Temporal Texture Modeling

by

Marcin Olof Szummer

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1995

Author ....
Department of Electrical Engineering and Computer Science
August 11, 1995

Certified by ....
Rosalind W. Picard
Associate Professor
Thesis Supervisor

Accepted by ....
F. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Temporal Texture Modeling

by

Marcin Olof Szummer

Submitted to the Department of Electrical Engineering and Computer Science on August 11, 1995, in partial fulfillment of the requirements for the degrees of Bachelor of Science in Computer Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Temporal textures are textures with motion. Examples include wavy water, rising steam and a crowd milling about. We model image sequences of temporal textures using the spatio-temporal autoregressive model (STAR). This model expresses each pixel as a linear combination of surrounding pixels lagged both in space and in time. The model provides a basis both for recognition and synthesis. We show how the least squares method can accurately estimate model parameters for large, causal neighborhoods with more than 1000 parameters. Synthesis results show that the model can adequately capture the spatial and temporal characteristics of many temporal textures.

Thesis Supervisor: Rosalind W. Picard
Title: Associate Professor

Temporal Texture Modeling

by

Marcin Olof Szummer

Submitted to the Department of Electrical Engineering and Computer Science
on August 11, 1995, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Temporal textures are textures with motion. Examples include wavy water, rising steam and a crowd milling about. We model image sequences of temporal textures using the spatio-temporal autoregressive model (STAR). This model expresses each pixel as a linear combination of surrounding pixels lagged both in space and in time. The model provides a basis both for recognition and synthesis. We show how the least squares method can accurately estimate model parameters for large, causal neighborhoods with more than 1000 parameters. Synthesis results show that the model can adequately capture the spatial and temporal characteristics of many temporal textures.

Thesis Supervisor: Rosalind W. Picard
Title: Associate Professor

Acknowledgments

I wish to express my deepest gratitude to prof. Rosalind Picard, who has been much more than just my thesis supervisor. Her imaginative suggestions, cheerful humor, encouragement and brilliant intellect has allowed me to become friends with the waves of the Charles, steam from the underground and other complex phenomena.

I am also grateful to Michael Grunkin. By sharing his expertise in texture modeling, he has helped me find the right path towards the goal, to focus on it and to avoid many stealthy pitfalls.

Kris Popat and Thomas Minka have always provided me with clear answers and challenging questions; while scrambling to find the answers, we have made new discoveries.

Many thanks to John Wang, Fang Liu and Greta Ljung.

The vision and modeling group at the MIT Media lab has been an exciting environment to work in. The colorful concentration of creativity and genius makes it a most dynamic and rewarding experience to indulge in. Thanks to you all!

To my parents,
Barbara and Maciej,
for all your love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What are temporal textures?

Temporal textures are textures with motion. Examples of temporal textures are wavy water, rising smoke, leaves or grass rippling in the wind, flocks of birds circling in the sky, or a crowd of people milling about. Temporal texture has been defined more precisely by Polana and Nelson [34] as motion patterns of indeterminate spatial and temporal extent. This is in contrast to *activities*, which are temporally periodic but spatially restricted. A person walking or swimming, or the spinning wheels of a machine, are examples of activities. A third class of motions are *motion events* which are single motions that do not repeat spatially or temporally. This class includes motions such as opening a door, lifting a briefcase, or throwing a ball.

An alternative characterization of temporal textures is based on the types of physical deformations occurring during the motion [1]. The simplest type of motion is *rigid motion*, which involves no deformation at all. All distances and angles of the moving object are preserved. This area has been extensively studied in the field of structure from motion, and other fields in machine vision. Rigid motion can be generalized by connecting several rigid objects together. Now, *articulated motion* is produced when the rigid parts move non-rigidly with respect to each other. This is useful for approximating human motion using stick figures. If we continue relaxing the constraints, we obtain *elastic motion*, which is any continuous, smooth motion allowing deformations. Many temporal textures are of this form. However, textures such as turbulent water and gases belong to the least restricted class of motions, called *fluid motion*.

The world is full of motion, and there are numerous phenomena that do not fall into any of the categories mentioned so far, but can still be treated as temporal textures. One example is motion induced textures. Even a rigid landscape moving with a simple translational motion can seem like a texture when the observer is sitting in a train and gazing through the window. Alternatively, walk through a forest and look towards the sky to see the glittering patterns of light finding their way through the leaf-work. Temporal textures can also be created simply by amassing objects: flocks of geese, swarms of bees, and popcorn popping in the oven.

In today's modern urban "deserts" and office landscapes, temporal textures are

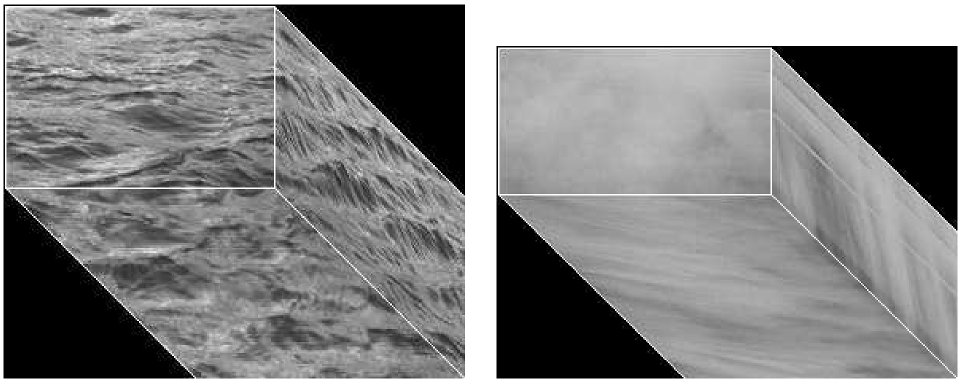

Figure 1-1: Two familiar temporal textures: wavy water and steam, displayed in an xyt volume.

less common than in nature. Perhaps it is the lack of natural motion that explains their dull, lifeless image? Nevertheless, technology has enabled us to create new, artificial temporal textures such as fountains, flags, fireworks and even indoor waterfalls. Human beings are fond of motion patterns, as judged by video games and screen savers showing wandering ants, schools of fish, explosions, or the evolving populations in Conway's game of life. Hopefully, tomorrow's environment will contain even more motion, more temporal textures, and be more stimulating to our perceptual system.

1.2 Modeling temporal textures

Temporal textures are prevalent in nature. How would a bird survive unless it could distinguish the wind-blown trees from an approaching predator? When we look out on a stormy ocean on a foggy day, and a chaos of rapidly changing, blurry grey light strikes our retina, what does the human visual system do? These questions have not been extensively studied and remain unanswered. Despite the omnipresence of temporal textures, most machine vision research regard them as "noise" that should be avoided, since many algorithms do not consider such cases, and as a consequence, fail miserably.

1.2.1 Objectives

In this thesis, the goal is to model temporal textures. We must find a representation that can capture both the spatial and temporal aspects of the textures. The model must be applicable for recognition tasks, such as identifying a given video sequence as belonging to the wavy water class. It should also be useful for segmentation, in other words, for partitioning a video sequence into homogeneous regions.

I believe that the spatio-temporal autoregressive model developed here is capable of representing a wide range of temporal textures. It models each pixel as a linear combination of neighboring pixels in space and time. It is a three-dimensional extension of autoregressive models that have been used with great success in one and two dimensions. The quality of the modeling can be judged by examining synthetic temporal textures produced by it.

1.3 Motivation

Motion and temporal textures are important from a perceptual point of view. Many animals are blind to anything that does not move. Motion is typically the first thing attended to in most perceptual systems. For some creatures certain motion patterns trigger specific actions. For example, divergent flow activates a landing response in houseflies [29]. It is probably easier for human beings to recognize an image sequence of a temporal texture, rather than to recognize a single still image of the same texture. In fact, still images are totally artificial in that they either require an apparatus that can capture light at an instant, or the interpretation of an artist. Thus, temporal textures pose a problem in their own right, and cannot be subsumed under still-image texture techniques.

From an applications point of view, the major motivation for working on temporal textures is the advent of large video databases. Hundreds of thousands of hours of video are available in digital form, and there are currently very few ways of retrieving a sequence based on its visual contents. Models of temporal textures are necessary for searching through such databases [33].

Nevertheless, there has been very little research on temporal textures in the context of recognition and segmentation. The reason is that such research was impossible only a few years ago, since video sequences contain orders of magnitude more data than single images. As a result, sequences require orders of magnitude more computational resources such as RAM memory, disk storage, processing power and bandwidth.

Today, the resources are available, and there is a continent of uncharted territory that is ripe for exploration. The texture field is very cross-disciplinary, drawing ideas from time series analysis, system identification, machine learning and neural networks, statistics, dynamical systems, image processing, machine vision and visual perception. Many of these disciplines have seen a boom in recent years. Research on the temporal texture field in particular will shed light on the closely related areas of three-dimensional texture modeling (such as medical imagery) and also on regular two-dimensional textures.

1.4 Applications

Applications of temporal texture research can be divided into the areas of recognition, segmentation and synthesis. Recognition tasks include the detection of specific textures, such as smoke or fire. Traditional smoke detectors are based on the physical or chemical properties of smoke. However, to prevent forest fires, we would like to monitor large areas remotely. With temporal texture models, we can detect smoke and fire using purely optical means by placing a camera on top of a hill or in a satellite. Another detection task can help to locate people in boat accidents, since it is difficult for humans to spot survivors in the sea. Other surveillance applications require that we ignore certain motions but attend to others. For example, waving trees should not set off burglar alarms. The most general recognition application is the video database query task mentioned previously. This task is difficult, since the image sequences are unconstrained and contain many things other than temporal textures. Some segmentation is required, and will be discussed next.

Segmentation can find the statistically similar regions in the video database query task. A natural decomposition of video is the layered representation used by Wang and Adelson [42]. The segmentation is based on clustering an affine flow field of the scene. Unfortunately, the flow fields of most temporal textures are very complicated, and will not cluster well. Instead of using the affine flow, we could handle such cases by clustering the temporal texture features extracted by the model.

Segmentation is also very useful for three-dimensional textures found in medical applications (note that temporal textures are three-dimensional textures with the third dimension set to time). When obtaining a computer tomography (CT) or magnetic resonance (MRI) scan of a brain, we would like to measure the volume occupied by different tissues or tumors. Another application is to measure the strength of bones from high-resolution CT scans to diagnose osteoporosis [11].

Apart from doing recognition and segmentation, we can also generate synthetic temporal texture sequences based on the model. Currently, computer graphics researchers spend great efforts on developing specialized models for particular textures such as fire [30]. A synthetic temporal texture model is much easier to construct: given a video clip of the desired texture, compute the texture parameters, and synthesize the result. A temporal texture model probably does not have as convenient "control knobs" as does a well-designed computer graphics model, so that changing the appearance of a synthetic fire would require trial and error. Nevertheless, the temporal texture model is perfect for applications such as data compression, where we want to recreate a given arbitrary texture with as few parameters as possible. Synthetic temporal textures would also be useful in design applications, such as for customizable screen-savers.

Chapter 2

Relevant Work

2.1 Introduction

Temporal texture research is based on ideas from many fields; perhaps the most relevant ones are machine vision, image processing, time series analysis and computer graphics. These fields have different approaches, goals and nomenclature. It is helpful to classify them according to whether the main objective is recognition, representation or synthesis of temporal textures.

- Recognition. This is often the goal in machine vision. The techniques are based on features optimized for discrimination rather than for representation, and they are usually insufficient for performing synthesis. Many techniques for image database query fall into this class.

- Representation. Statistical models, such as Markov random fields, model the interrelationships between pixels. Likewise, in time series analysis, the task is to discover the structure of the series and to predict future values.

- Synthesis. In computer graphics, the objective is to generate a realistic image sequence. The model need not be acquired automatically from video footage; it can be designed by hand for a particular texture. It is important that the model is intuitive to manipulate so that the desired look is obtained.

The spatio-temporal autoregressive model developed in this thesis falls into the representation category. Fortunately, there is significant overlap among the three goals. For example, models which can represent a texture well can usually be used to synthesize that texture. By virtue of preserving perceptually relevant information, this representation can be used for anything we could do using the original texture. Thus, recognition can be done either directly based on the parameters, or trivially by first synthesizing the represented texture and then performing recognition.

Techniques that aim to do recognition must be benchmarked especially carefully. Unless the data set is very large, the recognition algorithm may seem to perform well, but in fact may be recognizing the wrong attributes. The problem is that the algorithm is not grouping the textures according to perceptual qualities, but rather according to some coincidental similarities. Techniques that can be used both for recognition and synthesis (such as the autoregressive model) are less likely to fall into this trap, since the synthesis will reveal whether the model has captured the relevant perceptual qualities. Due to the current lack of large test sets of temporal textures (my test set is of size 10, and is the largest test set used so far), perfect recognition would be easy to achieve. A much more difficult goal is perceptually similar synthesis. Therefore, the evaluation in this thesis is based on synthesis. I will still describe the research done in the recognition aspects of temporal textures below.

2.2 Recognition of Motion and Temporal Textures

Researchers in machine vision have traditionally attempted to do object recognition using low-level features such as edges, junctions or points of maximum curvature. Temporal texture recognition is different from object recognition in that there may be a large number of small, independently moving parts, or no discernible parts at all. The low-level features used must be based on the statistics of a large number of points, rather than on individual edges. This is the approach followed by many of the methods below.

Another issue is the domain in which the features are extracted. Nelson and Polana [29] compute features based on the normal component of optical flow. Thus, their recognition is based solely on motion. They argue that visual signals contain more information than is necessary for recognition, and that from a functional point of view, as much information as possible should be discarded. This has the added advantage that the remaining signal is invariant to irrelevant properties, such as illumination or color. In the case of optical flow, the signal is also more directly meaningful than the original gray level input, since a flow vector (e.g. "downward motion") is more informative than a grey level value by itself. The disadvantage of basing the features on optical flow is that we may have discarded too much information. The optical flow is also inaccurate for most temporal textures, as it is based on the assumption of translating pixels that do not change in intensity, which is more accurate for object motion.

Nelson and Polana use four features on the normal component of the optical flow. The first feature measures the directional nonuniformity. A histogram describing the magnitude of motion in eight directions is determined, then its deviation from a uniform distribution is computed. The second feature measures the peakiness of the velocity distribution, and is simply the average motion magnitude divided by its standard deviation (also known as the *inverse coefficient of variation*). The third feature describes the distribution of directions. First, a cooccurrence matrix of four motion directions is computed, at a lag proportional to the average motion magnitude. In each direction, the number of pixel pairs that have a motion direction that differs by at most one is divided by the number of pixels that have a motion direction that differs by more than one. Four subfeatures are obtained. The fourth feature is the average positive and negative curl and divergence. In a recognition task with seven temporal textures, the third and fourth features achieve 100% discrimination by themselves.

One important assumption made in the above discussion is that the image has been segmented into homogeneous temporal texture regions. Segmentation becomes even more important when we characterize the motion of objects with determinate spatial extent. Usually, it is also necessary to normalize the size of the object and to normalize the time scale of the motion, so that matching with a model can proceed. Segmentation and normalization can be based on static image cues and be done independently of the motion characterization, but can also benefit from motion cues.

An example of segmentation and normalization using motion is given by Allmen and Dyer [2]. They find spatio-temporal curves that are tangent to the optical flow at each frame. Intuitively, these curves trace out the position of a specific point in space-time. They can be computed directly from the optical flow using the Runge-Kutta approximation method for differential equations. The points belonging to an object have interdependent motions, so the spatio-temporal curves are related. The slope and curvature is computed at each frame, and then curves with similar values are clustered. Each cluster is likely to correspond to an object. After obtaining this segmentation of the image, Allmen and Dyer propose to examine the curvature in scale-space. They desire to distinguish between the *relative* and *common* motion of the parts of an object. Unfortunately, only synthetic images were used in their experiment. In practice, it is difficult to track points through an image sequence, because of occlusion and noise.

Spatio-temporal curves can also be used to characterize periodic motions. Nelson and Polana [35] track objects using the centroid of the moving pixels in the frame. The size and the position of the object is normalized. The bounding box of the object is divided into square cells and the motion magnitude in each cell is summed. The idea is to use the motion magnitudes as features. If the motion is periodic, the motion magnitude of a cell will have a periodic pattern. For each cell, Fourier analysis is used to measure the degree of periodicity and the period of the motion. A feature vector is constructed from the motion magnitudes of the cells for several frames of a cycle. Different motions such as walking, running, and swinging can be recognized using a nearest centroid classifier.

Seitz and Dyer [38] suggest that purely temporal information can be used for the analysis of cyclic motion. The variation in the period can be characteristic for a motion, and can be described by the shortest and the longest cycle times, cyclic acceleration and points of irregularity. Seitz and Dyer demonstrate how to determine the instantaneous period, which is the interval from a given time t until the cycle reaches the same phase again. To determine whether the phase is the same in two frames, the mean-squared pixel intensity difference can be used if the view-point is fixed. This crude technique works for a sequence of X-rays of the heart. However, a view-point invariant match function must be used when the camera is moving.

Several other low-level characterizations of motion exist in the literature. Interesting motion events can be extracted from the spatio-temporal curves. For example, discontinuities of the curves and its derivatives indicate that the motion changed direction [8]. The spatio-temporal motion energy can be used as a feature [16, 3].

2.3 Representation

A multitude of statistical methods have been developed for spatial data; however, only a few of them have been generalized to spatio-temporal data. Below, we will review a few texture models including autoregressive models and a fractal model. For a more extensive survey, see [41, 15, 44, 40].

Autoregressive models have been used extensively for spatial data, and the most popular version is abbreviated SAR (simultaneous autoregression). Autoregressive models express each data point as a linear combination of neighboring points plus a white noise term. By allowing the noise to be correlated according to a linear combination of other values, we obtain the more general mixed autoregressive moving average model (ARMA). Mathematical definitions will be given in chapter 3. Here I would like to mention a modern variant of these models. Mao and Jain [27] suggest how to construct a rotationally invariant SAR model. Rotation invariance can be achieved by associating each parameter with a neighborhood that is averaged around a circle. They also construct a multi-scale model by estimating multiple SAR models. Each model has the same neighbor topology but the neighbors are at different distances from the predicted location. Unfortunately, the parameter estimation technique used by the authors is biased; specifically the least-squares estimator is applied for noncausal neighborhoods, which yields biased estimates [9].

ARMA models have been generalized for spatio-temporal data [6]. The generalization amounts to using spatio-temporal neighborhoods. For example, Pfeifer and Deutsch [32] model the number of arrests in 14 Boston police districts over 72 months. Each district has a neighbor computed as the average of the adjacent districts at the previous time step. The authors identify a spatio-temporal ARMA model by examining the autocorrelation and partial autocorrelation functions. Parameters are estimated using the Marquardt nonlinear optimization algorithm with a locally linearized least-squares cost function. In subsequent work [31], they study the autocorrelation function patterns for a few low-order models. Throughout, the neighborhoods include only elements from previous times; hence, there are no purely spatial neighbors. This restriction is an easy way to ensure causality and to create spatially symmetric neighborhoods. However, it is beneficial to include data from the same time step, since this data is usually highly correlated. In both studies, the authors use very small neighborhoods with only 1 or 2 parameters. This is perhaps adequate for prediction in their 1008 point data set. However, the requirements are quite different for image sequences with 10^6 to 10^7 pixels, and long distance correlations.

Heeger and Pentland [17] describe a fractal model for turbulent flow. Fractals are characterized by self-similarity across scales, which determines the fractal dimension. Turbulent phenomena such as clouds exhibit fractal scaling for ranges from .16 to 1000 km. The authors apply Gabor filters to determine the fractal scaling parameter (which is simply related to the fractal dimension). If the scaling parameter is consistent across different scales, then the region is likely to be a fractal. In this fashion, we can recognize turbulent flow, such as the wake behind a boat. Synthesis is also possible using Brownian fractals. Sequences of moving clouds, fluttering tree leaves and other turbulent flows have been generated with fractal models.

.

2.4 Synthesis

Fire and water are two temporal textures of great interest in computer graphics. Their uses include realistic flight simulators and special effects for the movie industry. General references are [43, 7].

Perry and Picard [30] present a modified particle system for fire synthesis. The particles are polygons made up of triangles with a common vertex, arranged to get a roughly elliptical shape. Each triangle is Gouraud shaded and can be semi-transparent. A flame is constructed by integrating the light from a single particle over a 25 step trajectory. The color, transparency and size of the triangles changes along the path. To create realistic dynamics, the convection currents and an external wind field are modeled. The synthetic fire model also incorporates material properties and will spread in a physically-based way. Other gaseous phenomena such as smoke and clouds can be simulated using particle systems as well [39, 5].

Kung and Richards [21] synthesize dynamic water. The water surface is modeled as a sum of squared sine waves in two dimensions, with slowly varying wavelengths. Motion is introduced by changing the phase of the sinusoids for different times. The sun is not included in the model. Instead, the illumination comes entirely from the sky, which is represented as a hemispheric light source. The reflectance function relies mainly on the Fresnel reflectance term. The goal of the authors is not just to perform synthesis, but also to understand how people infer water from an image. To this end, they describe a characteristic feature for water: "a very elongated, horizontal elliptical-like blob having cusped end-points at its extremal viewing positions." (p. 233).

Chapter 3

The STAR model

3.1 Introduction

Autoregressive models have been successfully fit not only for time series, but also for spatial data, such as images of textures. This thesis demonstrates that they can be used for spatio-temporal modeling as well.

I propose to model image sequences of moving textures using spatio-temporal autoregressive models (STAR). These models are three-dimensional versions of the regular AR model. Every pixel is modeled as a linear combination of neighboring pixels in time and space plus a noise term. Many techniques from one-dimensional AR models are applicable to three-dimensional models as well.

Here, I examine only linear models. Nonlinear models can be obtained by regressing each pixel on products of neighboring pixels. These nonlinear models can describe a wider class of phenomena, but are often very expensive computationally. In any case, it is good practice to try a linear model before proceeding to nonlinear models.

3.2 The Spatio-temporal Autoregressive Model

A linear autoregressive model (AR) has the form

$$s(\mathbf{x}) = \sum_{i=1}^{p} \phi_i s(\mathbf{x} + \mathbf{\Delta x_i}) + a(\mathbf{x}).$$

Originally, AR models were used to describe time series signals, in which case $s(x)$ is a one-dimensional signal. They were later extended to two dimensions [46]. In this context, $s(\mathbf{x})$ is a two-dimensional signal where $\mathbf{x} = (x, y)$. This model is often referred to as SAR, the simultaneous autoregressive model.

More recently, Cliff and Ord [6] introduced the spatio-temporal autoregressive model (STAR). This is simply the autoregressive model in space-time. For the STAR model, $\mathbf{x}$ is a vector denoting a location in space-time, i.e. $\mathbf{x} = (x, y, t)$, and $\mathbf{\Delta x} = (\Delta x, \Delta y, \Delta t)$. The term $a(\mathbf{x})$ denotes a white noise process, assumed to be Gaussian for parameter estimation purposes. Expanding out the vector notation, the model can be written

$$s(x,y,t) = \sum_{i=1}^{p} \phi_i s(x+\Delta x_i, y+\Delta y_i, t+\Delta t_i) + a(x,y,t).$$

The lags $\Delta x_i, \Delta y_i$ and Δt_i specify the neighbor structure of the model.

AR models can predict a given pixel from its neighbors. The one-step ahead forecast is given by

$$\hat{s}(\mathbf{x}) = \sum_{i=1}^{p} \phi_i s(\mathbf{x} + \mathbf{\Delta x_i}).$$

3.3 Autoregressive Moving Average Models

A more general model than STAR is the mixed spatio-temporal autoregressive and moving average model (STARMA). It is defined by

$$s(\mathbf{x}) = \sum_{i=1}^{p} \phi_i s(\mathbf{x} + \mathbf{\Delta x_i}) + \sum_{j=1}^{q} \theta_j a(\mathbf{x} + \mathbf{\Delta x_j}) + a(\mathbf{x}).$$

The ϕ_i are autoregressive parameters, and the θ_j are moving average parameters. As a special case, when $p = 0$ the model has only moving average terms and is called a spatio-temporal moving average process (STMA). Unfortunately, both the STMA and the STARMA model are nonlinear in the moving average parameters θ_j. Parameter estimation becomes difficult, especially considering that the number of parameters in a STARMA model is generally larger than for ARMA models. Hence, I have used the STAR model.

3.4 Properties of STAR

3.4.1 Causality

It is easier to synthesize and estimate parameters for causal models. For one-dimensional models, causality requires that the conditioning neighborhood consist only of data from earlier time steps. In higher dimensions, there are more ways to make a model causal. Causal STAR models have conditioning neighborhoods that are only a subset of the spatio-temporal volume. The neighborhood must correspond to a recursively computable filter. An example of a causal neighborhood is the nonsymmetric half-space (NSHS), such as the (x,y,t) subset defined by $t < 0 \vee (t = 0 \wedge y < 0) \vee (t = 0 \wedge y = 0 \wedge x < 0)$ (Figure 3-1).

The main advantage of causal models is that parameter estimation is easy. The conditional least squares method gives consistent estimates. In other words, the variance of the estimates goes asymptotically to zero as the number of data points increases. Unfortunately, least-squares does not have this property for non-causal models. In fact, the least squares estimates may have a large systematic bias which

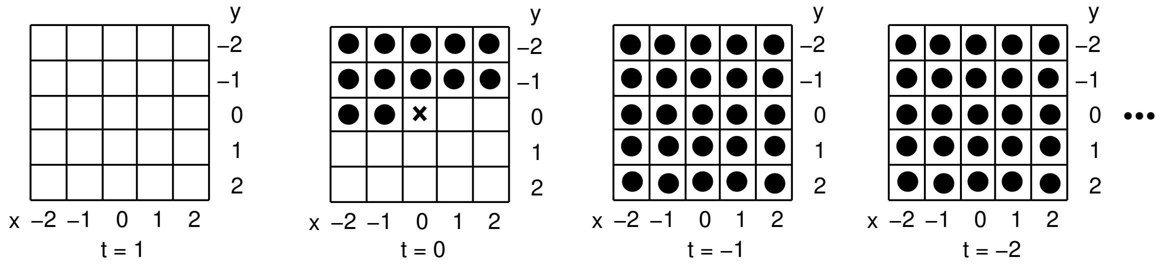

Figure 3-1: A nonsymmetric half-space neighborhood

is several times greater than the 95% confidence interval of the estimates [9, p. 125]. Hence, other techniques must be used for estimating parameters, and these involve nonlinear optimization, which is prohibitive computationally.

Another advantage of causal models is that synthesis is trivial, since the model is recursively computable. For noncausal models, synthesis requires solving a system of equations, which can be done using Gauss-Seidel relaxation. Alternatively, synthesis can be performed in the Fourier domain [18], or using Monte-Carlo techniques when there is an equivalent Gibbs random field.

The disadvantage of the causality constraint is that it is somewhat unnatural when used for spatial processes. It introduces an arbitrary directional bias, which depends on the orientation of the nonsymmetric half-space neighborhood (or any other causal neighborhood used). For spatio-temporal processes, the spatial asymmetry is not as severe as for purely spatial process. The spatial asymmetry arises only from restrictions for neighbors at $t = 0$, whereas neighbors at $t < 0$ can be symmetric. In fact, the spatial asymmetry can be completely eliminated by conditioning only on neighbors at $t < 0$. Thus, we can trade off spatial asymmetry against temporal asymmetry. Since time has a clear direction, and the physical world is believed to be causal, temporal asymmetry is easily justified.

In my experiments, I have used subsets of the nonsymmetric half-space neighborhood described above. While it has some directional bias, it exploits the strong correlations of neighbors at $t = 0$, which is especially important for temporal textures with fast motion.

3.4.2 Correlation Structure

The autocorrelation function (ACF) and the partial autocorrelation function (PACF) are useful tools for analyzing the correlation structure of autoregressive processes and for model identification [45]. For a STAR model, the ACF is defined by:

$$\rho(\Delta x, \Delta y, \Delta t) = \frac{\sum_{x=1}^{N_x - \Delta x} \sum_{y=1}^{N_y - \Delta y} \sum_{t=1}^{N_t - \Delta t} \dot{s}(x,y,t)\dot{s}(x+\Delta x, y+\Delta y, t+\Delta t)}{\sum_{x=1}^{N_x} \sum_{y=1}^{N_y} \sum_{t=1}^{N_t} \dot{s}(x,y,t)^2}$$

where $\dot{s} = s - E(s)$, and N_x, N_y, N_t are the (x, y, t) dimensions.

The PACF can also be extended to spatio-temporal processes, but is more compli-

Table 3.1: Theoretical ACF and PACF patterns for STARMA models

Model Form	ACF	PACF
STAR($\Delta x_{max}, \Delta y_{max}, \Delta t_{max}$)	tails off	cuts off after Δx_{max}, $\Delta y_{max}, \Delta t_{max}$ lags in x, y, t
STMA($\Delta x_{max}, \Delta y_{max}, \Delta t_{max}$)	cuts off after Δx_{max}, $\Delta y_{max}, \Delta t_{max}$ lags in x, y, t	tails off
STARMA($\Delta x_{max}, \Delta y_{max}, \Delta t_{max}$)	tails off	tails off

cated and involves solving a system of equations that is computationally expensive. Even the ACF takes a long time to calculate, because of the large number of data points that have to be multiplied to obtain an autocorrelation coefficient. It is faster to compute the ACF as the inverse Fourier transform of the power spectrum of the signal (Section 3.8).

The patterns of the ACF and the PACF for the spatio-temporal process are similar to their one-dimensional counterparts (Table 3.1, also see [31]). However, for a given image sequence, we can only compute sample estimates of the ACF and PACF that are noisy versions of the exact theoretical patterns.

3.5 Traditional Model Fitting for ARMA

> *The art of a time series analyst's model identification is very much like the method of an FBI agent's criminal search.* William Wei

To fit an ARMA model, we must decide on two things before we can start: the neighborhood size and its topology. Box and Jenkins [4] pioneered the following approach in the context of time series analysis.

1. Model identification: AR(p), MA(q), or ARMA(p,q)? Select model order (p and q).

2. Parameter estimation.

3. Model validation: is the model adequate?

To find a suitable model, the first step is always to plot the data. Since ARMA models can model only stationary processes, the next step is to transform the data to make it stationary. To stabilize the variance, we can take the logarithm or the square root of the series, or more generally apply the Box-Cox power transformation [45]. Then, to stabilize the mean, we (repeatedly) difference the series to produce a transformed series $s'(x) = s(x) - s(x-k)$. The value of k is often 1, and the differencing then corresponds to a high-pass filter with magnitude response $|H(e^{jw})| = |2\sin\omega/2|$. This operation removes slowly varying components (e.g. linearly increasing trends) from the data. For periodic sequences k is chosen to equal the period, which in the frequency domain is the filter $|H(e^{jw})| = |2 \sin k\omega/2|$. Such a filter cancels the periodic component of the signal.

There is another way to see why differencing stabilizes data with nonstationary mean. An unstable system has poles on or close to the unit circle. Differencing corresponds to multiplying the system by $(1 - z^{-k})$, which introduces a zero that cancels a pole on the unit circle.

To determine good values of p and q in the ARMA(p,q) model, the autocorrelation and partial autocorrelation functions are examined. When the data comes from a pure moving average ($p = 0$) or a pure autoregressive ($q = 0$) this decision is relatively easy, since the ACF or the PACF will have only one spike respectively. Unfortunately, when the data comes from a mixed ARMA process, the patterns of the ACF and PACF are complicated and the identification is unclear. In this case it is best to guess a model, estimate parameters for it, and check how well it fits the data.

The next step, parameter estimation, can often be done using least squares and is discussed more in section 3.7.

Finally, the model is checked by examining the residual errors, $e(x) = s(x) - \hat{s}(x)$, where $s(x)$ is the original series and $\hat{s}(x)$ is the predicted series. If the residuals look like white noise (examine the residual ACF and PACF), then the model is a good fit.

3.6 Model Fitting for STAR

The Box-Jenkins modeling methodology can be applied to identify STARMA models as well (see [32]). However, the conditioning neighborhoods for STAR models are much larger and can take on many shapes in space-time. Thus, the patterns of the ACF and PACF are complex and can only give weak hints about the model. In this section, I will first describe how to detect and remove nonstationarities. Next, I will suggest how to select a good neighborhood for a STAR model.

3.6.1 Detecting and removing nonstationarity

Data whose mean and variance are not constant over the spatio-temporal volume is called *nonstationary*. Many nonstationarities can be spotted by looking at the data. Often, the lighting conditions are not uniform across the image. For example, there can be an illumination gradient across the image, or certain surfaces may be tilted and reflect more light. Slow periodicity that cannot be directly captured by the model (e.g. periods much longer than the model neighborhood) can also be regarded as a nonstationarity, since the mean usually varies significantly within the period.

The ACF can give us more quantitative information. It aids in determining whether the data is stationary, and whether it is periodic. If the ACF falls off slowly (e.g. decays linearly rather than exponentially), the data is nonstationary and should be stabilized.

Unsharp masking is a useful technique to remove some nonstationarities [9]. One approach is to median-filter the image and subtract the filter output from the image

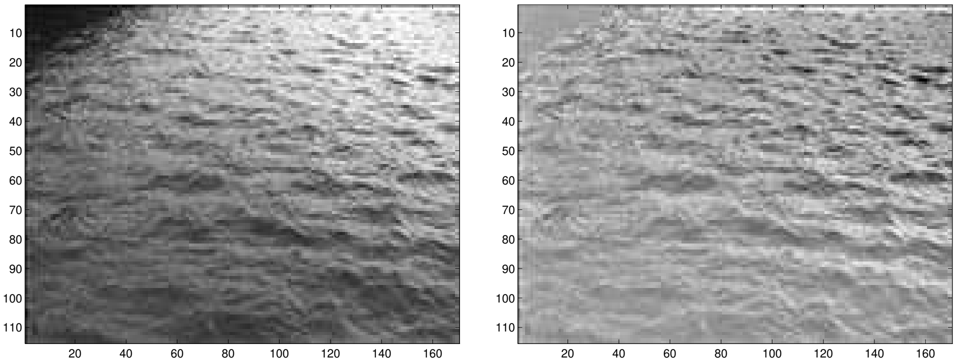

Figure 3-2: First frame of `river-far`. Original (left) and with local medians subtracted off (right).

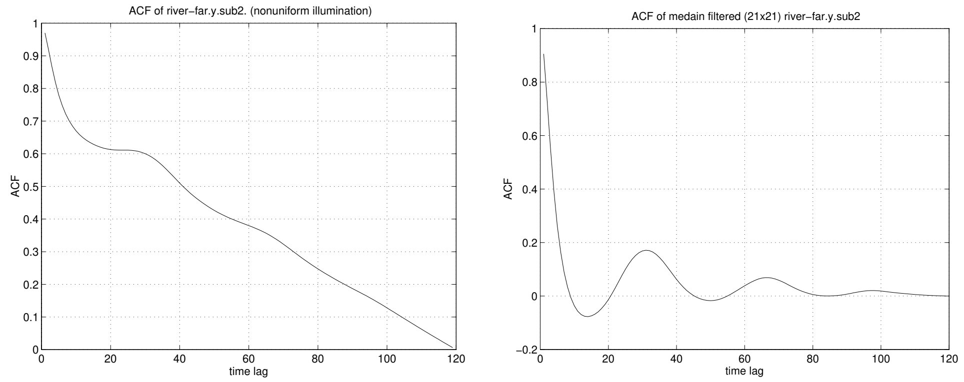

Figure 3-3: Temporal ACF of `river-far`. Original (left) and with local medians subtracted off (right).

(Figs. 3-2, 3-3). The ACF of the unfiltered river image falls off very slowly, indicating nonstationarity. This is due to an intensity gradient in the image—the upper portion of the image is brighter than the lower portion. To remove the nonstationarity, the output of a 21×21 spatial median filter is subtracted from each image in the sequence. This removes low-frequency illumination variations. The intensity gradient disappears, and the ACF now rapidly goes to zero.

For one-dimensional signals, differencing the data is the most common technique for removing nonstationarities. However, for spatio-temporal signals, differencing is more complicated, since now there are several possibilities: $s'(\mathbf{x}) = s(\mathbf{x}) - s(\mathbf{x} - \mathbf{\Delta x})$ where the direction of differencing $\mathbf{\Delta x}$ must be determined. The obvious choices are differencing along the time axis, or the x or y axes. Unfortunately, differencing along one axis may lead to overdifferencing along the other axes[1]. Moreover, since images have mostly low-frequency content, differencing with $|\mathbf{\Delta x}| = 1$ tends to remove too much information and leave mostly high-frequency noise. Thus, we should filter the signal more gently.

[1]This effect is due to *cointegration*. Series at given (x, y) locations in the volume are individually nonstationary, but some linear combination of them may be stationary [14].

In fact, we need not difference the series ourselves; instead, we can let the parameter estimation routine handle the nonstationarity. A nonstationary signal has poles on or close to the unit circle. The conditional least squares estimator will simply estimate an autoregressive spectrum that matches the spectrum of the signal, and if we use a large enough model, the poles at the unit circle will be well-represented. However, the exact maximum likelihood estimate cannot handle such poles and will give nonsensical answers. In this case we must cancel the poles by introducing matching zeros through differencing.

While differencing adjacent pixels ($|\mathbf{\Delta x}| = 1$) is not necessary, seasonal differencing ($|\mathbf{\Delta x}| > 1$) may be useful. For example, the river ACF has peaks at time lags 32, 65 and 97, which is periodic except for some estimation error. Seasonal differencing $s'(x, y, t) = s(x, y, t) - s(x, y, t - 32)$ might be appropriate. The parameter estimation algorithm could not exploit this periodicity, unless we used a very large neighborhood including 32 time lags or a very different model.

3.6.2 Neighbor pruning

For image sequence data, the sample ACF and PACF have complex patterns that cannot be easily interpreted. The IACF (inverse autocorrelation function) and the ESACF (extended sample autocorrelation function) [45] have the same problem. Thus, we cannot rely on them for determining model order, or for selecting the neighbors that should be included in the model.

Instead, we begin by fitting a very large model. We include all parameters that could possibly be important. Many of the parameters of this model will turn out to have values close to zero, and are not statistically significant. This is no cause for alarm, and the resulting model is already useful.

The model can be made more parsimonious by pruning insignificant parameters. The fewer parameters the model has, the lower is the variance of the remaining parameter estimates. Large models also need to be trained on more data to avoid overfitting, and require more computational resources. Grunkin [9] presents an algorithm that iteratively discards the least significant parameters while ensuring that the Schwartz's Bayesian Criterion (SBC) decreases. The SBC is an approximation to the Bayes' information criterion (BIC), which in turn is an updated version of the Akaike's AIC. Let $|\Omega|$ be the data set size, p be the neighborhood size, and $\hat{\sigma}_a^2$ be the estimated innovation variance. Then

$$\mathrm{SBC} = |\Omega| \ln \hat{\sigma}_a^2 + p \ln |\Omega|.$$

$$\mathrm{AIC} = |\Omega| \ln \hat{\sigma}_a^2 + 2p.$$

The significance of a parameter is determined by the t-test (the parameter value divided by its standard deviation). For static image textures, the pruning algorithm typically reduces 80 parameter models to 50 parameters while maintaining the visual quality of the simulated texture [9].

3.7 Parameter Estimation

There are three main ways to perform parameter estimation for ARMA models.

- Conditional maximum likelihood. For causal ARMA models, equals conditional least squares (CLS).
- Unconditional maximum likelihood. For causal ARMA models, equals unconditional least squares (ULS).
- Exact maximum likelihood (ML).

The ARMA model requires a neighborhood of pixels to predict a given pixel. Near the boundaries, some neighbors are missing. The three methods differ in the assumptions about the boundaries. The conditional technique assume a specific value for the boundary (see Section 3.7.1). The unconditional method attempts to predict the missing values using backcasting. Backcasting involves applying the ARMA model in the reverse direction [4]. Finally, the exact maximum likelihood makes no assumptions about the boundary at all.

The exact maximum likelihood method is regarded to be the most accurate, but is difficult to derive and has no closed form solution for the estimates. The unconditional method is an approximation to the exact method, because it assumes that the missing values can be predicted according to an ARMA model. The boundary effects are insignificant if the data has large dimensions (such as $100 \times 100 \times 100$) compared to the model, and the data is not seasonal or close to being nonstationary. Most visual textures are nearly nonstationary. Nevertheless, since I have plenty of data, I will only use the conditional least squares method (CLS). This method is described below.

The least squares method attempts to minimize the total square prediction error E:

$$\begin{aligned} \min_{\hat{\phi}} E &= \min_{\hat{\phi}} \sum_{\mathbf{x}\in\Omega} e(\mathbf{x})^2 \\ &= \min_{\hat{\phi}} \sum_{\mathbf{x}\in\Omega} \left(s(\mathbf{x}) - \sum_{i=1}^{p} \hat{\phi}_i s(\mathbf{x} + \mathbf{\Delta x_i}) \right)^2 . \end{aligned} \tag{3.1}$$

To find the minimum, set the partial derivatives $\partial/\partial\phi_i$ of the last expression equal to zero for $i = 1, 2, \ldots, p$. After simplification, we get the p *normal equations*

$$\begin{bmatrix} R(1,1) & R(1,2) & \cdots & R(1,p) \\ R(1,2) & R(2,2) & \cdots & R(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ R(1,p) & R(2,p) & \vdots & R(p,p) \end{bmatrix} \begin{bmatrix} \hat{\phi}_1 \\ \hat{\phi}_2 \\ \vdots \\ \hat{\phi}_p \end{bmatrix} = \begin{bmatrix} R(0,1) \\ R(0,2) \\ \vdots \\ R(0,p) \end{bmatrix} \tag{3.2}$$

where

$$R(i,j) = \sum_{\mathbf{x}\in\Omega} s(\mathbf{x} + \mathbf{\Delta x_i}) s(\mathbf{x} + \mathbf{\Delta x_j}).$$

The parameters are now estimated by solving this linear system of equations. Let $\mathbf{A}$ be the matrix on the left containing covariances. $\mathbf{A}$ is a covariance matrix, and therefore symmetric positive semi-definite (we can safely assume it is positive definite, since the probability that the determinant equals 0 is small). The system can be solved efficiently using Cholesky decomposition $\mathbf{A} = \mathbf{Q}\mathbf{Q}^{\mathbf{T}}$.

If the model is large, the system of equations is often ill-conditioned. There exist methods called *square-root algorithms* that are numerically more stable. They avoid forming products of the data and instead try to directly obtain a matrix $\mathbf{Q}$ such that $\mathbf{Q}\mathbf{Q}^{\mathbf{T}} = \mathbf{A}$ (see Ljung [25, p. 275]).

The variance of the innovation sequence $a(\mathbf{x})$ can be estimated from

$$\hat{\sigma}_a^2 = \frac{E}{|\Omega| - p}.$$

The $|\Omega|$ is the number of data points used in the estimate. There is some debate about whether or not the number of estimated parameters p should be subtracted off in the denominator [4]. In the cases considered here, the difference is minimal since $|\Omega| \gg p$.

The total mean square error can be efficiently calculated (without looping through the data again) by

$$E = \sum_{\mathbf{x}\in\Omega} e(\mathbf{x})^2 = \sum_{\mathbf{x}\in\Omega} s(\mathbf{x})^2 - \sum_{i=1}^{p} \hat{\phi}_i \sum_{\mathbf{x}\in\Omega} s(\mathbf{x} + \Delta\mathbf{x}_i)s(\mathbf{x}) = R(0,0) - \sum_{i=1}^{p} \hat{\phi}_i R(0,i).$$

Assuming that the underlying process is well-modeled by the autoregressive model, then the approximate variance of the estimates is given by

$$Cov(\phi_1, \phi_2) \approx \hat{\sigma}_a^2 \begin{bmatrix} R(1,1) & R(1,2) & \cdots & R(1,p) \\ R(1,2) & R(2,2) & \cdots & R(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ R(1,p) & R(2,p) & \vdots & R(p,p) \end{bmatrix}^{-1}. \tag{3.3}$$

However, care must be taken with this expression. If we are uncertain whether the data is well described by an autoregressive process, it may be better to estimate the parameter covariance by dividing the data in subblocks, and estimate the parameters of each subblock separately. The parameter covariance can now be computed from the set of subblock estimates.

3.7.1 Treatment of boundaries: correlation and covariance methods

Several different assumptions can be made about the missing edge pixels when using the conditional maximum likelihood estimator. Three of the most common methods are:

- Correlation method: assume that the missing pixels have values equal to the mean of the data (usually 0).

- Covariance method: use only the inner portion of the data, so that all neighborhoods are contained in the data.

- Toroidal boundary assumption

In the previous section, we did not specify what the treatment of boundaries was. In fact, the above equations are equally valid for both the correlation and the covariance method. The two differ only in what region is used to minimize parameters. For the correlation method, the minimization is done over all Euclidean space, i.e. $\Omega = \mathbb{R}^n$. At the locations where actual data is not available, we assume $s(\mathbf{x}) = E[s(\mathbf{x})] = 0$ (the data is always made zero-mean before estimation). The covariance method, on the other hand, only uses existing data. Thus, Ω is the inner subset of the data such that the neighbors of every point in Ω are known.

The covariance method is recommended as more robust by Box and Jenkins [4]. However, the correlation method offers a few computational advantages. Firstly, the $R(i,j)$ are scaled autocorrelations, which can be computed as the inverse Fourier transform of the power spectrum of the data. This is much faster for models with many parameters (see Section 3.8). Secondly, for one-dimensional processes with contiguous neighborhoods $R(i,j)$ depends only on the difference of i and j, so we can write $R(i,j) = R(|i-j|)$. In this case, the covariance matrix in eq. (3.2) is a Toeplitz matrix. The system of equations can then be solved very efficiently using the Durbin-Levinson recursion [25]. Unfortunately, in higher dimensions we have that

$$R(i_1, j_1) = R(i_2, j_2) \text{ if and only if } \mathbf{\Delta x_{i_1}} - \mathbf{\Delta x_{j_1}} = \mathbf{\Delta x_{i_2}} - \mathbf{\Delta x_{j_2}}.$$

For the matrix to be Toeplitz we require that $\mathbf{\Delta x_{i_1}} - \mathbf{\Delta x_{j_1}} = \mathbf{\Delta x_{i_2}} - \mathbf{\Delta x_{j_2}}$ whenever $i_1 - j_1 = i_2 - j_2$. This condition is satisfied if and only if all the neighbor offsets Δx_i lie in a one-dimensional subspace. This is rarely the case and we must solve the system of equations some other way.

The covariance method appears to give better results in practice. We can examine the accuracy of a given procedure in the following manner:

1. Estimate parameters for a sequence.

2. Synthesize texture based on these parameters.

3. Estimate parameters for the synthesized volume. Compare with estimates from step 1.

A good parameter estimation routine should yield the same parameter estimates in steps 1 and 3. Table 3.7.1 shows the accuracy of the covariance and correlation methods on the river sequence. The covariance method estimates have relative errors of less than 1 percent for all but the smallest two parameters. The correlation method gives errors that are several times larger. Even worse, these errors are many times

Table 3.2: Accuracy of correlation and covariance method on sequence `river.y.sub2` using an approximately third order causal neighborhood with 10 parameters. Columns 2 and 3 show percent relative error = 100×error/parameter value. Columns 4 and 5 show error divided by standard deviation of the parameter estimates.

Actual parameters	Covariance % rel. error	Correlation % rel. error	Covariance error/std error	Correlation error/std error
0.1776	-0.3	7.9	-0.5	15.7
-0.0384	-0.5	1.6	-0.7	-2.0
1.0167	0.0	-2.2	0.0	-37.3
-0.2686	0.0	-8.1	0.0	43.6
-0.0766	0.7	14.1	-0.8	-18.0
-0.0071	-5.6	42.2	0.7	-5.0
0.3251	-0.4	7.8	-1.4	28.3
-0.0439	-0.2	-2.5	0.3	3.7
-0.1431	-0.3	13.4	0.8	-32.0
0.0296	2.7	-27.0	1.3	13.3

greater than the standard deviation of the estimates, as shown by the fourth column. Thus, the correlation method is not recommended.

In addition to the boundary assumptions assumed by the covariance method and the autocorrelation method, the toroidal boundary assumption is occasionally used. The data is assumed to wrap around so that $s(x, y, t) = s(x \text{mod} N_x, y \text{mod} N_y, t \text{mod} N_t)$ where data has size $N_x \times N_y \times N_t$. This assumption is mostly useful for simplifying the estimation of noncausal autoregressive models [19].

3.8 Practical Parameter Estimation

The most time consuming step of parameter estimation is the computation of correlations. When large amounts of data are involved, correlations can be evaluated much more efficiently in the Fourier domain than in the spatio-temporal domain. This is done as follows:

1. Zero pad the data $s(\mathbf{x})$ to double its dimensions, rounding up to the nearest power of 2.
2. Fourier transform $S(\omega) = \text{FFT}(s(\mathbf{x}))$.
3. Power spectrum $P(\omega) = |S(\omega)|^2$.
4. Inverse Fourier transform $R(\mathbf{k}) = \text{IFFT}(P(\omega))$.

Now, the ACF is obtained by normalizing $\rho(\mathbf{k}) = R(\mathbf{k})/R(\mathbf{0})$. The correlations are immediately ready for the autocorrelation method of parameter estimation (Section 3.7.1), and the notations in the two sections are reconciled by letting $R(i, j) = R(\mathbf{\Delta x_i} - \mathbf{\Delta x_j})$.

The data is zero padded in order to get a Fourier transform of size $2N_x \times 2N_y \times 2N_t$, assuming the original data dimensions are $N_x \times N_y \times N_t$. Padding is necessary to obtain a power spectrum whose inverse Fourier transform has $N_x \times N_y \times N_t$ distinct values, so that we can obtain all correlations (the power spectrum is real, even, and nonnegative, so that its inverse Fourier transform is purely real and even).

The efficiency of this method is due to the Fast Fourier Transform. For a one-dimensional signal, the FFT requires only $O(N \log N)$ computations to compute all correlations of an N point signal whereas a direct calculation would require $O(N^2)$ time. However, this is only true if N is a power of 2, or has a lot of small prime factors. It is usually advisable to round N up to the nearest power of 2 to get good performance from the FFT algorithm.

For a spatio-temporal signal, we must calculate a three-dimensional FFT. The FFT is a separable computation, so we simply take one-dimensional FFTs along the x,y and t axes [22]. The run time is $O(N_x N_y N_t \log(N_x N_y N_t))$ which is a huge improvement compared to the direct computation $O(N_x^2 N_y^2 N_t^2)$. In practice, all autocorrelations of a $60 \times 85 \times 120$ sequence can be computed in 15 minutes on a 100 MFLOP computer. However, the required 3D Fourier transform of this sequence is of size 128 Mbytes (using double precision numbers), so it is necessary to have a large amount of RAM and disk space available.

3.9 Synthesis

Synthesis for causal STAR models is straightforward. First, the boundaries of the volume are initialized. Here, Gaussian random noise is used, but almost anything will do, since the initial conditions die off gradually. For the synthesis, each pixel is predicted as a linear combination of its (already synthesized) neighbors and of Gaussian random noise with the same variance as the innovation process $a(\mathbf{x})$. Next, the borders of the synthesized volume are cropped off to remove transients from the initial values. It is enough to crop off twice the largest neighborhood offset in every dimension.

At this point we have a synthesized floating point volume. To view the result, the image must be quantized to match the 256 grey-levels of the screen. An autoregressive process with a zero-mean Gaussian innovation process always produces volumes with zero-mean, Gaussian histograms. This is because a linear combination of Gaussians is still Gaussian. Unfortunately, this does not always correspond well to the histogram of the original sequence. We should also ensure that the synthesized volume has mean and variance similar to the original. An elegant solution is to match the histogram of the synthesized volume to the original. This step guarantees that the mean and variance of the two volumes is the same. A remarkable improvement in visual similarity is achieved for some textures such as steam (Figure 3-4) when this histogram matching is done.

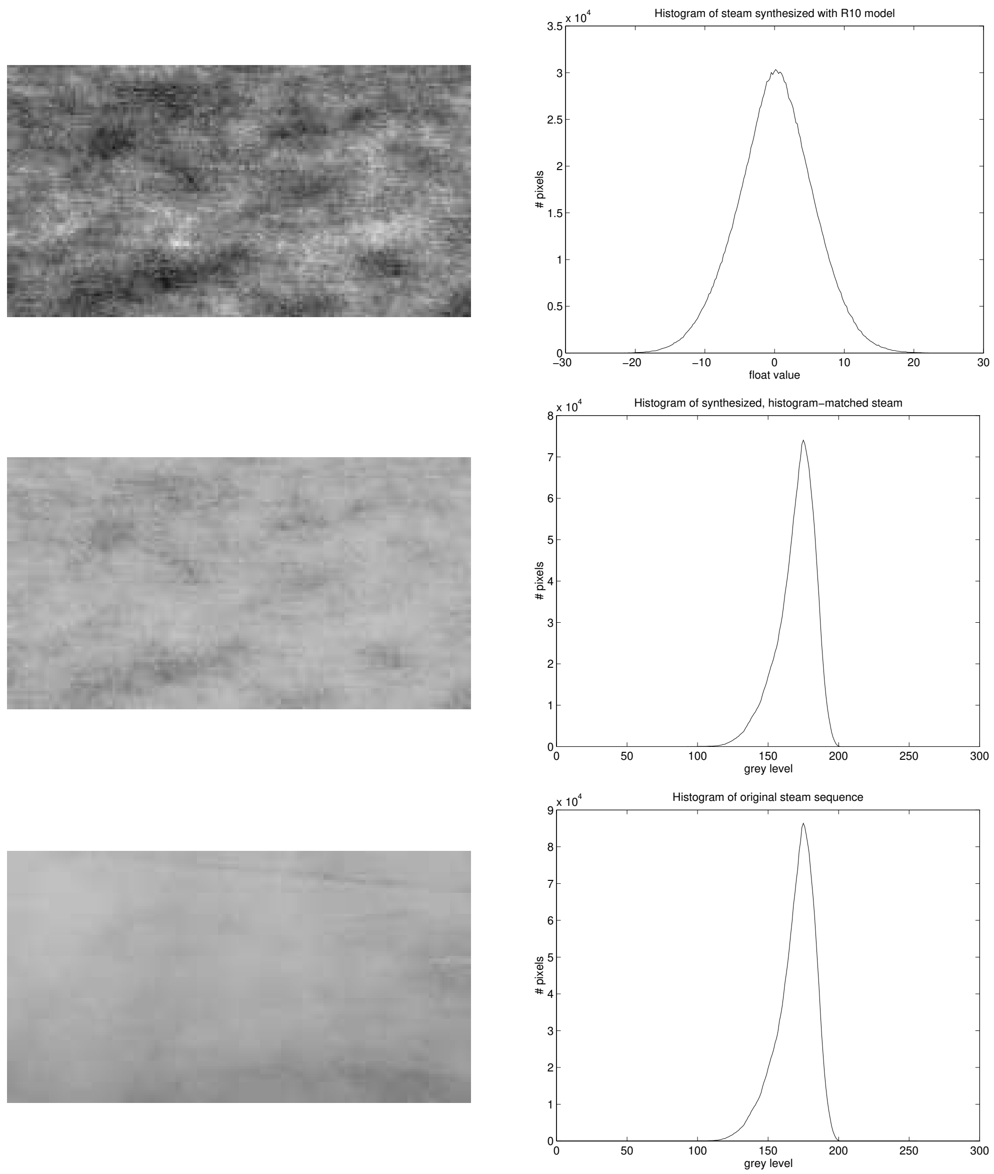

Figure 3-4: Histogram matching improves the visual similarity between synthesized and original texture. Synthesized texture (top), synthesized and histogram matched texture (middle) and original (bottom), with corresponding histograms in the right column.

Chapter 4

Results

4.1 The Data

The data consists of commonly occurring temporal textures (Table 4.1). The particular sequences were selected to get interesting, every-day indoor and outdoor textures with different motions. With a few exceptions, the chosen textures appear to be wide sense stationary and amenable to statistical texture techniques. The data was acquired by me to avoid copyright issues, and will be made available to other researchers.

The image sequences of temporal textures were recorded using a Hi-8 video camera. A tripod was used to avoid camera motion. The sequences were digitized at a resolution of 720×487, Y-B-R (8-4-4 bit) color and 30 frames per second onto D1 digital format. Next, the sequences were cropped to only contain temporal textures, then deinterlaced, low-pass filtered and subsampled. Only the luminance information was used. The final resolution in all cases was about 170×115 and the length was 120 frames (corresponding to 4 seconds). Thus, each sequence contains about 2 million data points.

A few special challenges are present in the data. Fluid surfaces are highly specular, and the characteristic highlights are difficult to predict without any representation of the surface geometry. Some textures extend towards the horizon, and show perspective chirping. The swirling water motion in the toilet sequence is highly non-

Table 4.1: Image sequences used

Name	Contents	Challenges
`river-near`	close-up of water	large motion
`river-far`	wide-angle shot of water	specularity, perspective chirp
`steam`	steam from manhole	
`boil-heavy`	vigorously boiling water	
`boil-light`	lightly boiling water	specularity
`plastic`	windblown plastic sheet	
`toilet`	swirling water in toilet	spiral motion

stationary. On the whole, none of the textures is perfectly wide-sense stationary. The illumination and motion are always more or less nonuniform across space and time. To reduce the illumination variations, the output of a 21×21 spatial median filter was subtracted from the data sequences.

4.2 Autocorrelation functions

The spatio-temporal autocorrelation function for the `river-near` sequence is very interesting (Figure 4-1). The x-t slice through the ACF shows periodicity through time. For small x-offsets, the ACF has local maxima at time offsets 32, 65 and 97. I believe that the periodicity is due to the recurring waves in the image sequence. A time lag of 32 corresponds to an actual time of 1.1 seconds, which is a reasonable wave period.

The y-t slice shows repeated diagonal streaks of correlation. In the image sequence, the waves are moving from the top of the image towards the bottom (along the y-axis). This motion explains the diagonal correlations in the y-t plane. After approximately 32 time lags, a new wave appears with strong correlations at $y = 0$.

Surprisingly, there is no visible periodicity in the purely spatial correlations (the x-y slice). The correlations are strong along the horizontal x-axis, but decay rapidly along the y-axis. This agrees with the fact that waves are approximately horizontal. In the `river-near` sequence, only one period is visible at a time, and this explains why we do not see any vertical periodicity in the x-y slice.

For the `river-far` sequence we see similar behavior. Since the waves are smaller, more periods are visible, and vertical periodicity in the purely spatial direction can be seen.

Taken altogether, the autocorrelation functions show that the signals have structure both in the spatial and temporal dimensions simultaneously. Thus, full spatio-temporal modeling is necessary to capture all aspects of the signals. Purely spatial or purely temporal modeling is not sufficient. Another observation is that the sequences have *long memory*, in other words, that the correlations can be significant even for large offsets in space and time.

4.3 Parameter estimates

The parameters were estimated in the following fashion. The least-squares estimator using the covariance method (computed directly in the spatio-temporal domain without FFTs) was used to construct the normal equations. The Cholesky decomposition for symmetric positive definite matrices was applied to solve the system of normal equations, and to invert them to obtain the covariance matrix of the estimates.

The causal neighborhoods used in the experiments are summarized in Table 4.2 and a few are depicted in Figure 4-2. Note that the number of neighbors is very large compared to what is commonly used for one and two-dimensional autoregressive models. There are hundreds, even more than a thousand parameters. However, recall that the data sets used for parameter estimation have dimensions $170 \times 115 \times 120$,

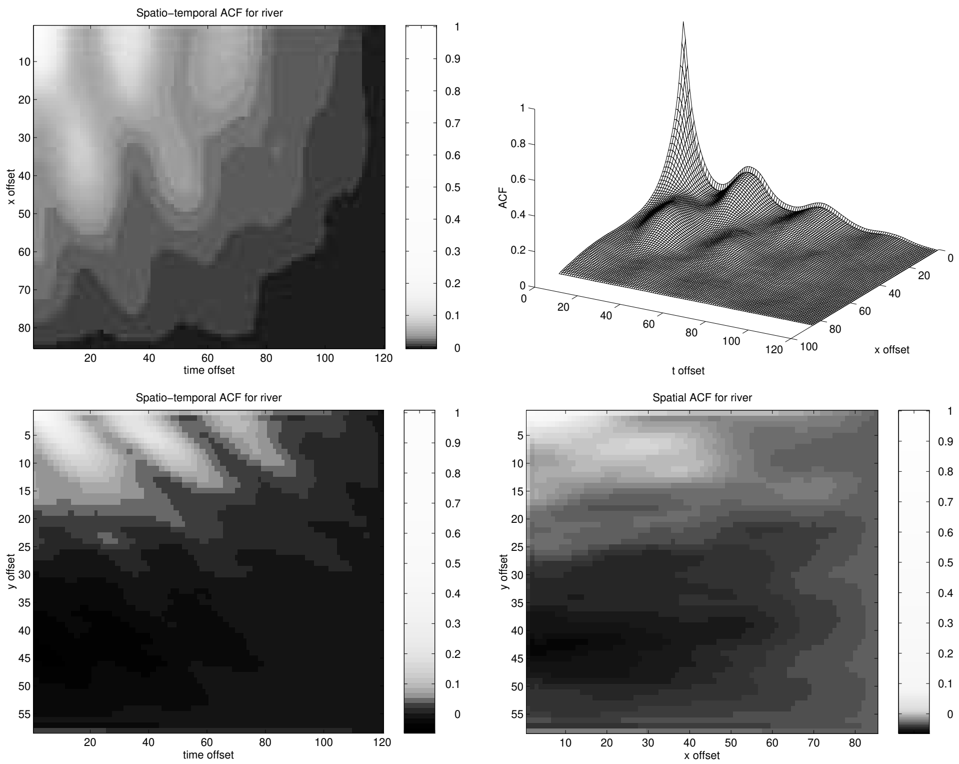

Figure 4-1: Autocorrelation function for `river-near`. An x-t slice (top two graphs), an y-t slice (bottom left) and an x-y slice (bottom right). All slices are at 0 offset in the fixed dimension.

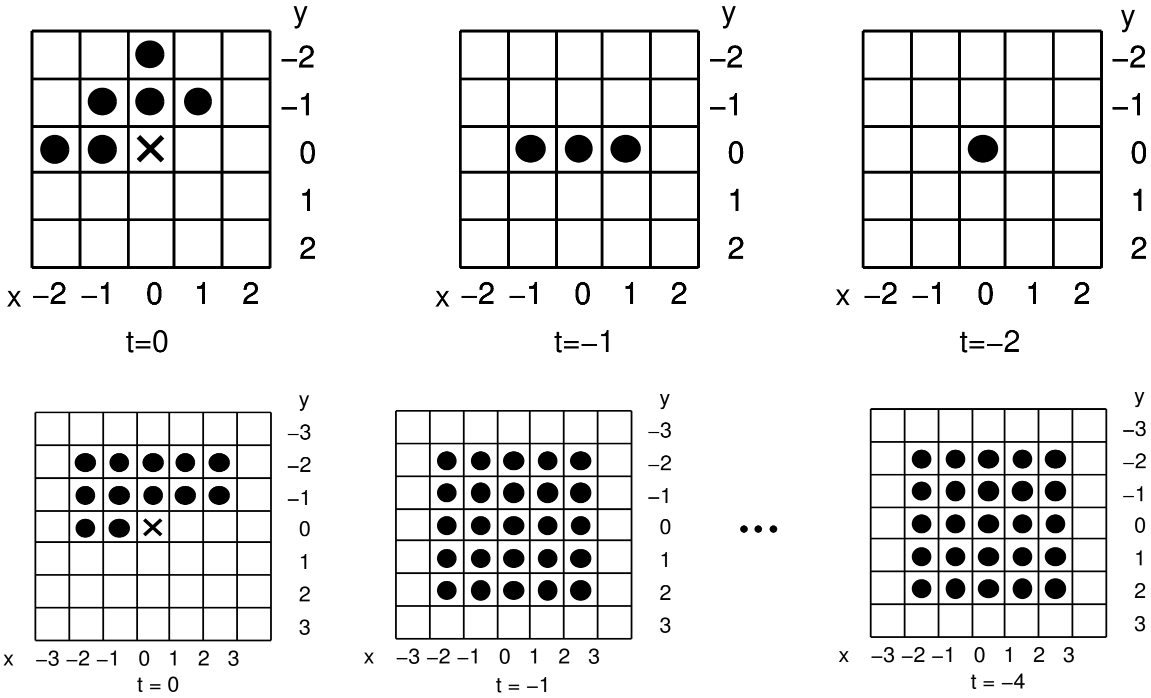

Figure 4-2: Neighborhoods `n10` (top; 10 parameters) and `B5` (bottom; 112 parameters).

Table 4.2: Neighborhoods

Neighborhood	No. neighbors	Shape
`n10`	10	10 closest pixels
`C4`	128	half-sphere, radius 4
`C7`	709	half-sphere, radius 7
`B11`	1270	box, approx. $11 \times 11 \times 11$

and contain more than 2 million data points. Thus, there are about 2000 data points per parameter in the `B11` model, reducing the risk for overfitting. The large number of neighbors is simply the consequence of modeling all three dimensions, rather than one or two. However, we must take precautions to ensure that all parameters can be reliably estimated.

4.4 Estimation accuracy

When working with very large models, it is important to make sure that the algorithms can accurately estimate the parameters. Unless the algorithms are numerically stable, the results may be meaningless.

The parameter estimation involves solving the system of normal equations (eq. 3.2), which becomes ill-conditioned as the number of parameters increases. Figure 4-3

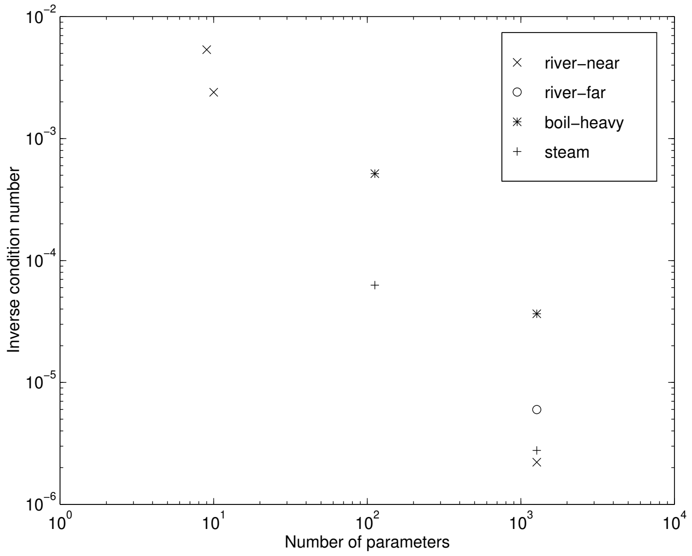

Figure 4-3: Inverse condition numbers for normal equations matrix.

shows the inverse condition numbers of the matrix for the normal equations. The inverse condition number is on the order of 10^{-6} for the `B11` models. Fortunately, this is well within the limits of double precision arithmetic which typically has 16 significant digits. However, single precision floating point arithmetic would not give accurate results.

Perhaps the best measurement of the estimation accuracy of the algorithm is to run it on synthesized sequences with known parameters. Section 3.7.1 described such a test for a 10 parameter model. The relative error of the covariance method estimates was very good, less than 1% except for the two parameters with magnitudes close to zero.

A similar test has been made for the `B11` model with 1270 parameters. Only 440 of the 1270 parameters were statistically significant, in other words had magnitudes at least two times larger than their standard deviations. For insignificant parameters, the relative error can be large. However, the relative error for the majority of the statistically significant parameters is less than 25% (Figure 4-4). The relative error for the estimate of the innovation variance is 0.01%.

The algorithm computes not only the parameters themselves, but also the variance and covariance of the parameters (equation 3.3). This information is very useful for recognition applications, where we weight the parameters inversely to their errors and take their covariance into account (Mahalanobis distance). Moreover, parameter significance can be determined using a t-test, which is the ratio between the parameter value and its standard deviation. Thus, we desire that these covariances be accurate as well.

For the 10 parameter model applied to `river-near`, all parameter values lay within two standard deviations of the parameter estimates. The actual variance of the parameter estimates can be computed by dividing the data set into subblocks. Such

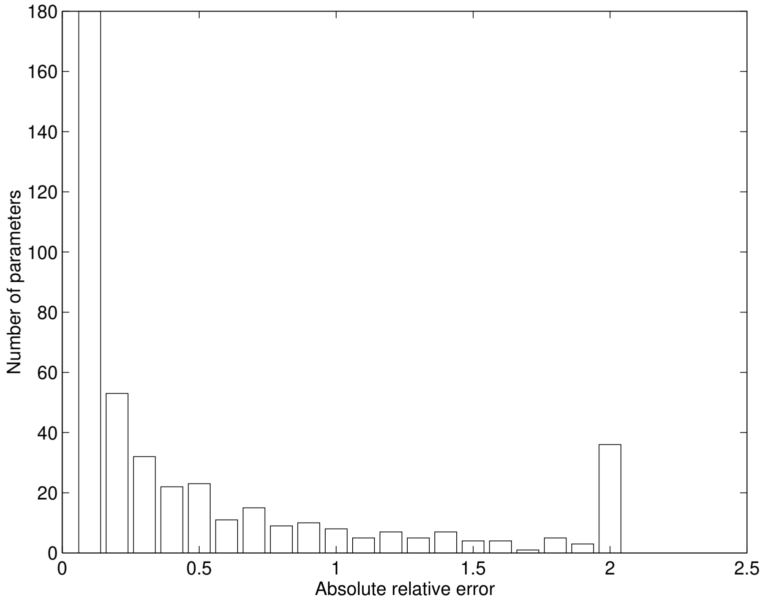

Figure 4-4: Absolute relative error of the 440 significant parameters for `B11` model of `river-far`. The relative error is the ratio (actual–estimated) / actual value. The rightmost bar includes all parameters that have relative errors 2 or larger.

an experiment shows that the parameter estimates have an approximately Gaussian distribution (Figure 4-5). This is what we expect from linear regression theory for stationary processes [32], and it suggests that the texture is approximately stationary.

The actual variance is very close to the variance predicted by the algorithm (Table 4.3). The variance of the estimates is approximately inversely proportional to the amount of data. So, if we divide the data into 64 subblocks, the parameter estimates given by one subblock have 64 times larger variance than the estimates given by the whole volume. Interestingly, the average of the 27 or the 64 subblock estimates is just as accurate as the estimate obtained from the whole volume. This observation may be useful for recognition applications, where we can quickly get approximate texture parameters from a single subblock. Then, if the approximate texture parameters are interesting we can obtain successively more accurate parameters by averaging more subblocks.

4.5 Synthesized sequences

The synthesis results are presented in Figures 4-6 and 4-7. Large neighborhoods (128–1270 parameters) have been used. The perceptual quality of some textures is very good; especially `steam` and `boil-heavy` are very convincing, both in terms of the motion and the spatial texture. The model cannot reproduce the bubbles in the boiling water, however. The two river sequences are not quite as well modeled, but are clearly recognizable as water. The difficulty is the specularity of fluids, which is not captured by the model. As a result, the water looks too grainy. Moreover, the

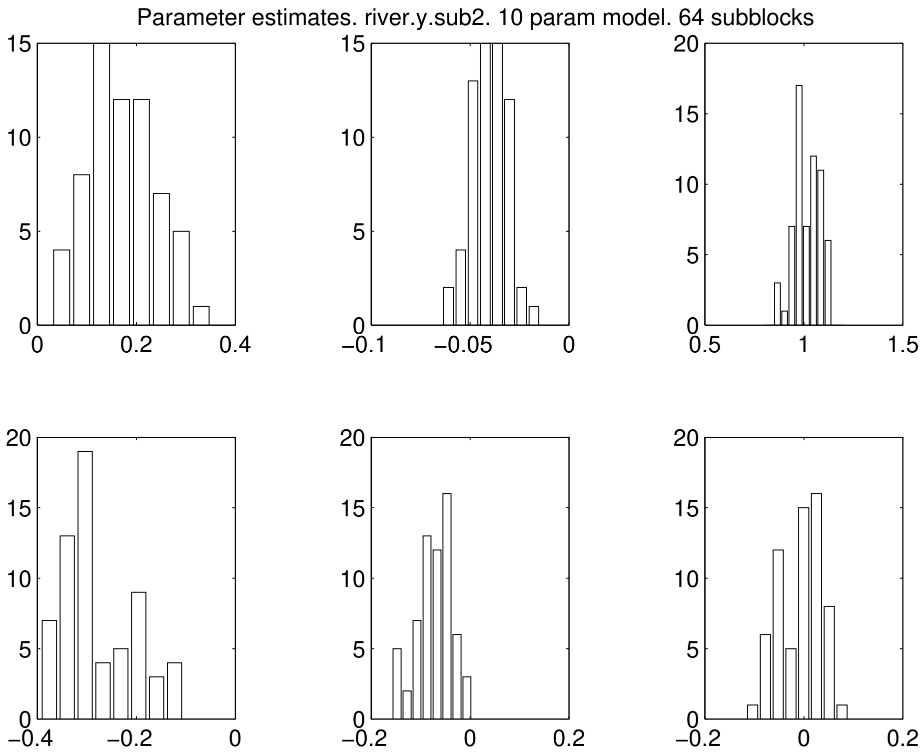

Figure 4-5: Distribution of parameter estimates for 64 subblocks of `river-near` using neighborhood `n10`. The parameters correspond to the first six parameters of table 3.7.1.

Table 4.3: Actual and predicted standard error for parameter estimates for 27 subblocks and 64 subblocks.

27 subblocks		64 subblocks	
actual std err	estimated std err	actual std err	estimated std err
0.0049	0.0049	0.0083	0.0079
0.0014	0.0014	0.0024	0.0024
0.0038	0.0038	0.0051	0.0055
0.0034	0.0034	0.0047	0.0048
0.0031	0.0031	0.0052	0.0052
0.0033	0.0033	0.0053	0.0052
0.0051	0.0051	0.0072	0.0079
0.0026	0.0026	0.0026	0.0030
0.0038	0.0038	0.0050	0.0057
0.0036	0.0036	0.0056	0.0053

original `river-far` has perspective chirp. The STAR model averages out the chirp to produce a middle frequency.

The remaining sequences are more challenging. Lightly boiling water `boil-light` has extreme specularity. Successive frames have very different appearances, even though they are only separated by 1/30 of a second. The synthesized still frames look very different from the originals, but when played back, the same shimmering is seen. The windblown plastic sheet `plastic` shows the opposite behavior. In this case, the texture of the still images is modeled well, but the occasional single back-and-forth wave is synthesized as a sea of undulating wrinkles. Finally, the spiraling motion in `toilet` cannot be represented at all by the autoregressive model, and instead produces a texture that looks like blurred random noise.

The optical flow of the synthesized sequences is similar to the optical flow of the originals (Fig. 4-8 and 4-9). However, the optical flow for temporal textures is not very accurate, since it assumes constant brightness which is often violated due to specularities. For example, the optical flow for the `toilet` sequence does not show any spiraling motion.

4.6 Discussion

4.6.1 Why use STAR for temporal texture?

The spatio-temporal autoregressive model is one modeling technique out of many. In this section, I will examine some advantages and disadvantages of using STAR and autoregressive modeling in general and for temporal textures in particular.

Perhaps the main advantage of the STAR model is its simplicity. Every pixel is modeled as a linear combination of its neighbors plus white noise. Thus, the model is easy to analyze, and not surprisingly, the properties of autoregressive models are well understood. There is also a well-established procedure (Box-Jenkins modeling) for identifying ARMA models, estimating parameters and validating the models. Arguably, this procedure does not generalize to spatial or spatio-temporal processes, but it still provides a helpful foundation.

The STAR model is computationally efficient, which is important considering that each dataset consists of 10^6 to 10^7 points. The main steps involve computing correlations and solving a system of linear equations. These computations could be easily implemented in hardware for additional speed.

More importantly, autoregressive models are widely applicable in practice. They can obviously model signals with autoregressive spectra (spectra with poles, but no zeros). They can also approximate general spectra containing zeros, although a mixed autoregressive moving average model would be more parsimonious for this case. Wold's theorem states that any wide sense stationary, zero-mean process that has no linearly deterministic component (a linear combination of past values) can be represented by a moving average process [23, 24]. If the moving average process is invertible (has all zeros inside the unit sphere) it can be expressed in an autoregressive form. Of course, these are theoretical results that allow representations with infinitely

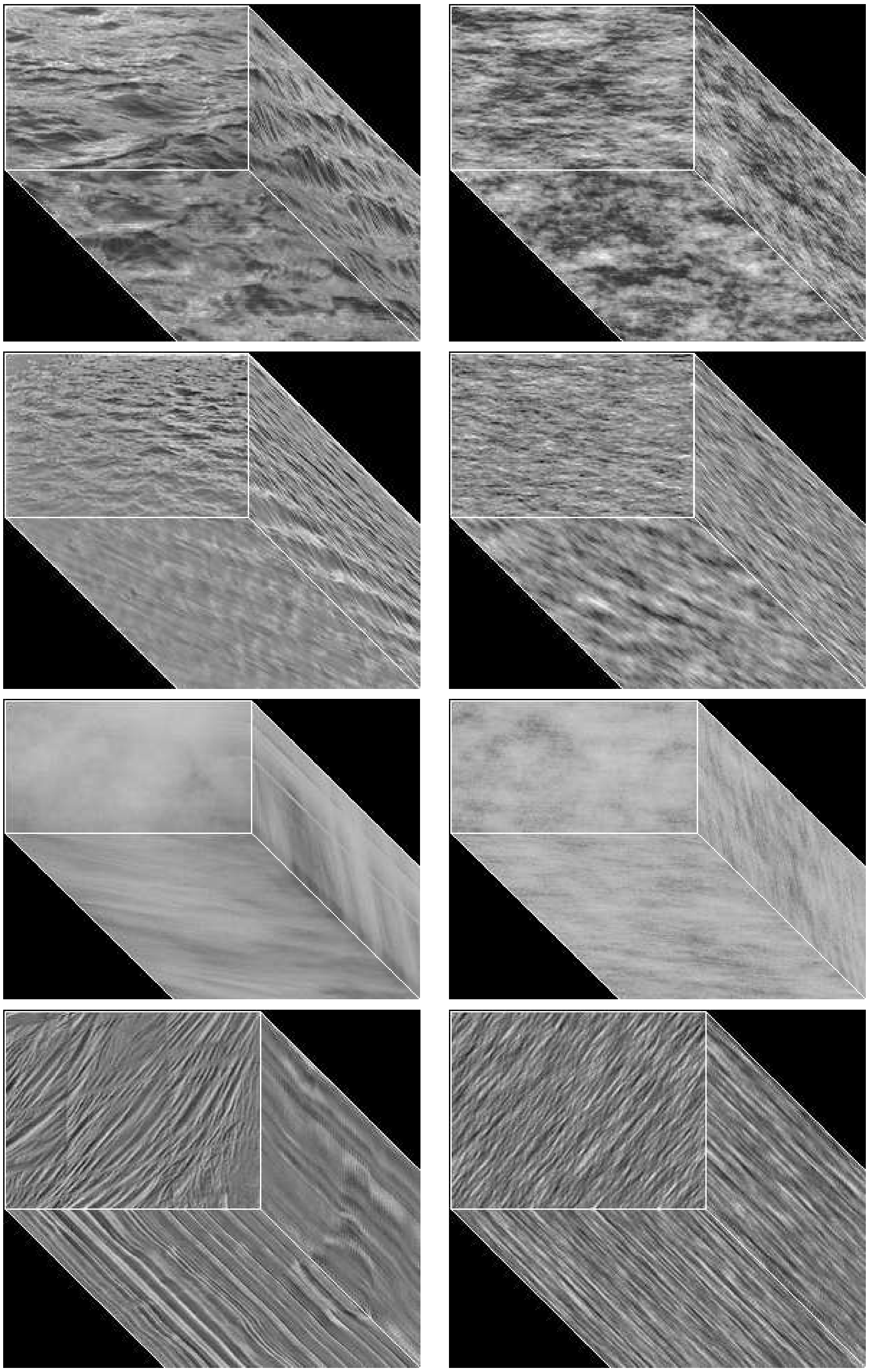

Figure 4-6: Synthesis results. Originals (left column) and synthesized (right). The sequences and neighborhoods are `river-near (B11)`, `river-far (B11)`, `steam (B11)`, `plastic (C7)`.

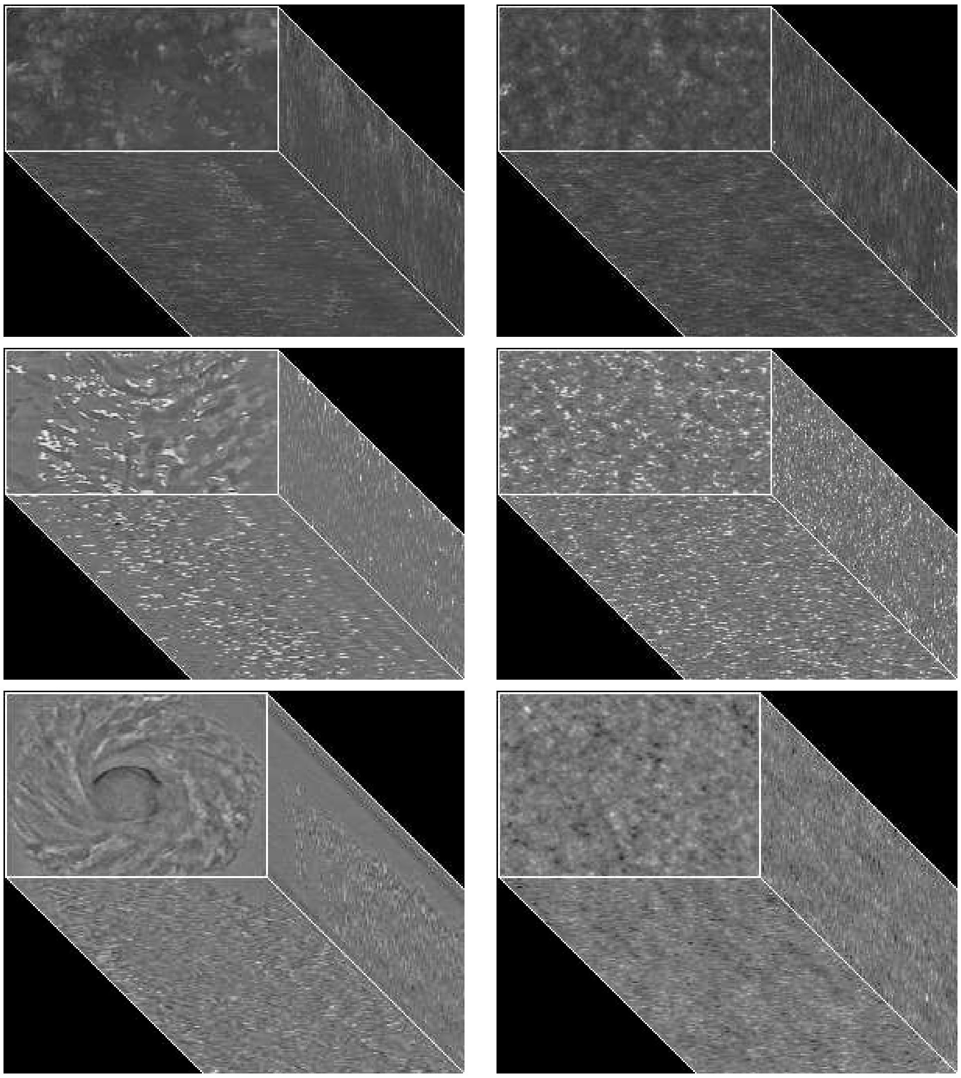

Figure 4-7: Synthesis results. Originals (left column) and synthesized (right). The sequences and neighborhoods are `boil-heavy`, `boil-light`, `toilet`.

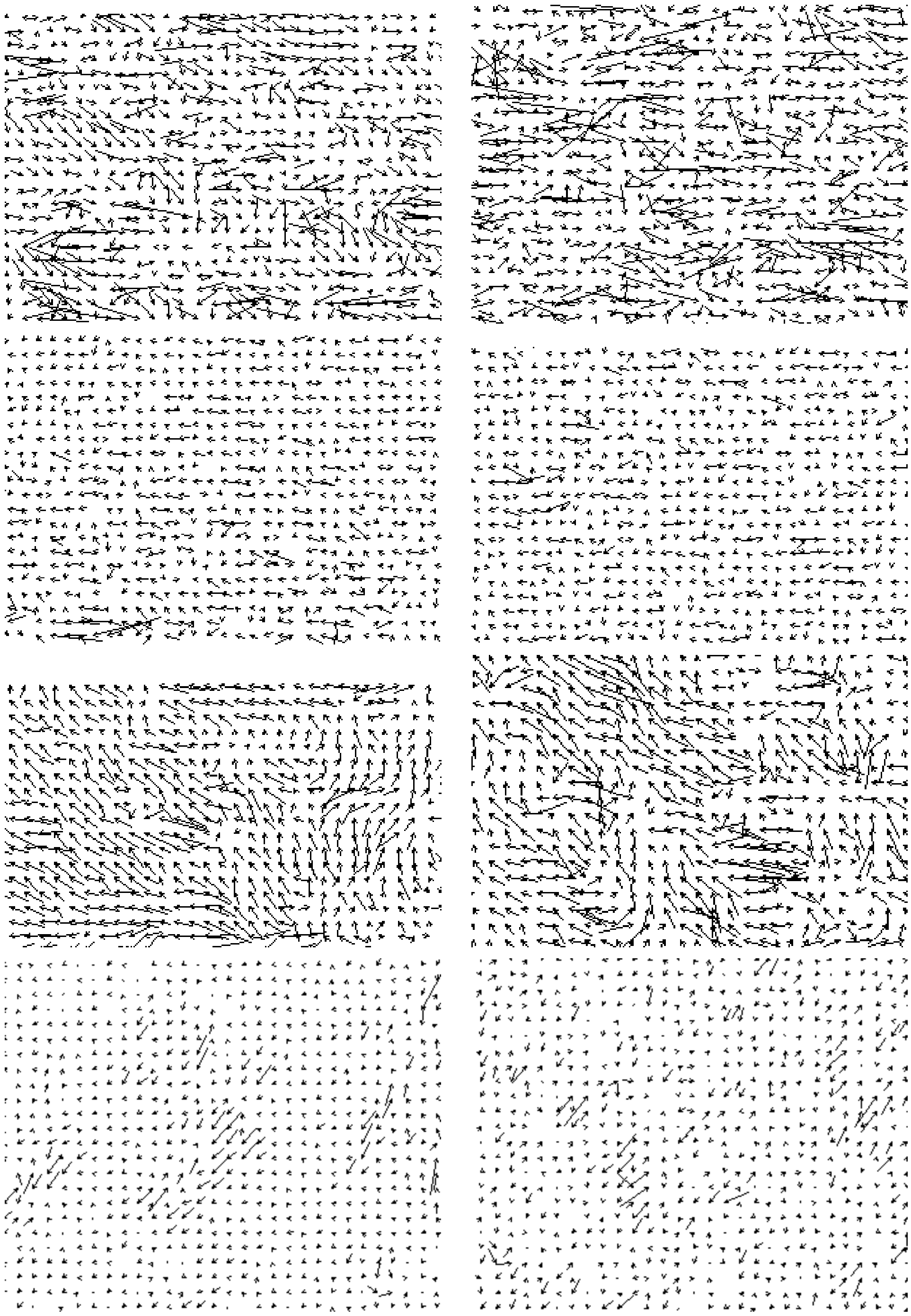

Figure 4-8: Optical flow of real image sequences using the Lucas-Kanade algorithm (left column) and synthesized sequences (right column). Sequences are `river-near` (top), `river-far`, `steam` and `plastic` (bottom).

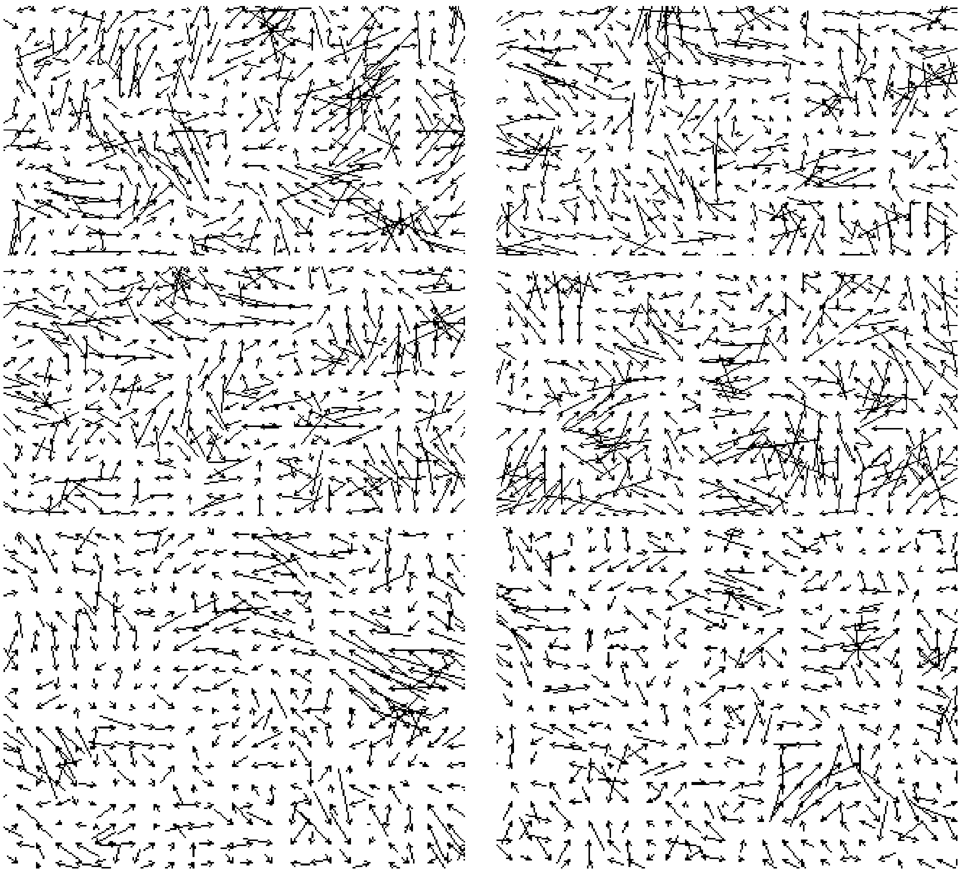

Figure 4-9: Optical flow of real image sequences using the Lucas-Kanade algorithm (left column) and synthesized sequences (right column). Sequences are `boil-heavy` (top), `boil-light` and `toilet` (bottom).

many terms.

A rule of thumb may be that autoregressive modeling is applicable anywhere a Fourier transform would make sense. Just like the Fourier transform, the autoregressive model is readily extended into an arbitrary number of dimensions, and many types of data.

The power and convenience offered by autoregressive modeling has made it popular in many contexts. Autoregressive models form the backbone of the fields of time-series analysis [45] and system-identification [25]. They are routinely used for speech modeling both for coding and recognition [37]. They have been used extensively for texture modeling of static images [27]. In a recent comparison of recognition performance on the Brodatz texture album [24], the best results were achieved by the Wold decomposition, whose core is an autoregressive model.

4.6.2 Limitations of the STAR model

The assumption of linearity lies at the heart of the autoregressive models. It reduces the complexity of the parameter estimation and the amount of data needed for modeling. Unfortunately, it fundamentally limits what this model can represent. If the relationship between pixels is not approximately linear, then the STAR model cannot capture it. Moreover, the autoregressive model can only represent wide-sense stationary data. In the context of temporal texture, rotational motion and expansion/contraction are not stationary, and cannot directly be modeled by STAR. Empirical evidence for this fact is given by the poor synthesis of the `toilet` sequence. More generally, any form of acceleration cannot be represented by STAR, since it is required that the velocity is the same everywhere (Fig. 4-10).

If an image sequence contains multiple textures, it is important to segment it into stationary regions, since the autoregressive model has no concept of multimodal distributions. It assumes that the texture is the same everywhere and will compute an average across the region.

4.7 Conclusion

I have modeled temporal textures using the spatio-temporal autoregressive model. This model offers several advantages over those used previously. Firstly, it can be used both for recognition and synthesis. Compared to pure recognition techniques, the model enables synthesis as a way to verify whether the data could be adequately captured. Compared to most techniques used in computer graphics, the model can be acquired automatically from video data.

The feasibility of using very large spatio-temporal neighborhoods (1000 parameters or more) has been demonstrated. The estimation accuracy has been empirically verified. Large neighborhoods are shown to be necessary to satisfactorily model the behavior of many temporal textures.

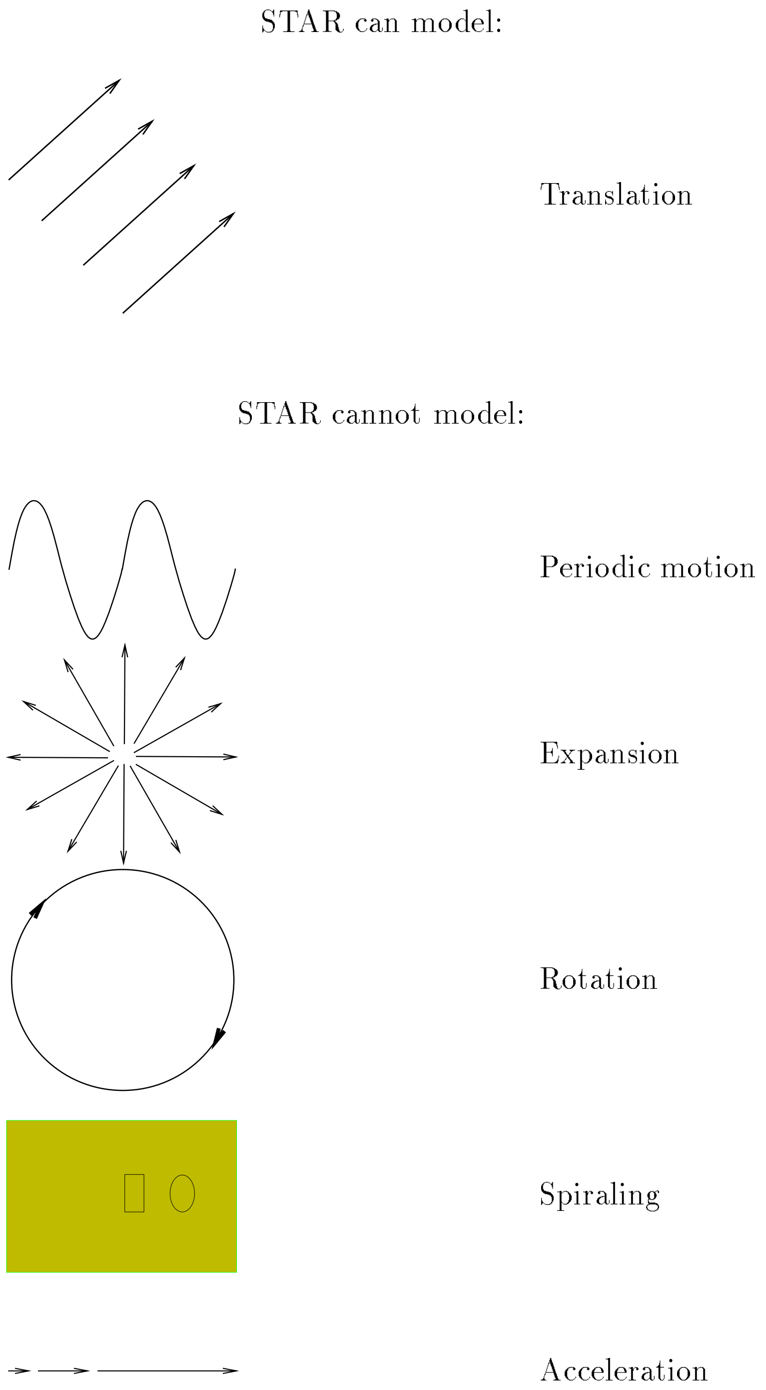

Figure 4-10: Six canonical motion patterns. The homogenous STAR can presently model translation.

4.8 Future Work

4.8.1 Recognition and segmentation

STAR coefficients can be used for recognition of temporal textures. First, compute the STAR coefficients and their covariances. It is beneficial to use several neighborhoods at different scales, as in the multi-resolution SAR technique [27]. Collect all these coefficients in a feature vector. Next, the similarity of two textures can be measured with the Mahalanobis metric. Note that this recognition technique is not invariant to the direction or speed of the motion. For example, water flowing slowly to the left will be different from water flowing rapidly to the right. A certain degree of motion invariance can be achieved by using neighborhoods based on spatio-temporal averages, in the same way that rotation invariance is obtained in the circular symmetric autoregressive model [20]. Unfortunately, the model may become too invariant and no longer be able to capture important differences.

Segmentation of temporal textures can be done by computing STAR feature vectors for small, overlapping windows of the volume, and then clustering these vectors. Segmentation is very expensive computationally, since parameters must be estimated for all the windows. Some speed-up can be gained from exploiting the overlap of windows.

4.8.2 Modeling other data types with the STARMA model

The STARMA model is a general three-dimensional ARMA model, and can be used for data other than image sequences. By renaming the t-axis to be the z-axis, we obtain a three-dimensional texture model. Such a model has a wealth of applications in the medical field, where three-dimensional data is widespread due to the success of computer tomography, MRI and other imaging techniques.

Alternatively, we can use the three-dimensional model for representing color textures. A standard color texture has three channels, RGB. First, we transform the texture to an appropriate colorspace, perhaps to decorrelate the color channels, or to use a more perceptual color space such as MTM [28]. Next, let the STAR t-axis become the channel axis. Each channel can now be modeled separately by regressing on all the channels. Note that this is no longer an *auto*regressive model, since the relationships between the channels are not symmetric. My initial experiments indicate that this model has better recognition performance than autoregressive models that only use the luminance information.

Instead of applying STAR to an image sequence, one can attempt to run STAR on the optical flow of the sequence. Optical flow explicitly represents motion. It may be easier to model motion directly based on optical flow, rather than indirectly based on luminance or color information.

Another straightforward extension is to model 4 dimensions instead of 3. During medical treatment of schizophrenia, Alzheimer and other diseases, one may wish to monitor the changes in the brain. Our data is four-dimensional, namely a time sequence of three-dimensional volumes.

4.8.3 Nonlinear models

Nonlinear models can represent a wider class of signals than can linear models. They may also require fewer parameters to describe some textures. We have initiated temporal texture modeling with the cluster-based probability model [36]. This is a powerful, nonlinear model capable of modeling high-dimensional probability distributions. Due to the curse of dimensionality, the model cannot handle neighborhoods of size more than 20. Fortunately, river water synthesized with only 10 parameters already looks fairly realistic. To model larger structures, a hierarchical analysis and synthesis technique will be used.

Appendix A

List of Temporal Textures

Flock of sheep
School of fish
Tadpoles
Flies
Bees
Ants
Termites
Swarm of mosquitoes
Seagulls
Reindeer horns on herd of reindeer

A mingling crowd of people
A pack of bicyclists

Waves
Wake from a ship
Boiling water
Boiling stew
Food processor blending fruit
Coffee beans being ground
Nuts being crushed

Rotating fan
Propellers
Rotating wheels
Rotating drum of washing machine
Sowing seeds
Blowing dust

Waterfalls
Fountains
Firehoses
Lawn sprinklers

Water rotating into drain

Fire
Smoke
Candles
Flickering lights

Flapping Sails
Flags
Laundry in the wind

Rain
Hail
Snow
Snow storms
Sand storms
Avalanches
Tornadoes
Hurricanes
Wind-tunnel experiments

Atom bomb clouds
Explosions
Fireworks
Video games

Text scrolling on screen
Printing presses (big sheets of paper)

Motion induced textures
Landscape while driving a car
Driving by a fence

Noise on TV screen
Underwater weeds in the tide

Twinkling stars
Mirages
The corona of the sun
Northern lights *(Aurora Borealis)*

Boiling spaghetti
Serving peas on a plate

Motion of blood cells, sperms
Hair on a windy day
Drops falling into water

Vibrating guitar string
Piano clubs moving

Pouring milk into water
Shower

Dissolves in television
Ice crystals forming
Glass shattering

Bibliography

[1] J. K. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Articulated and elastic non-rigid motion: A review. In *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 2–14, Austin, TX, 1994.

[2] Mark Allmen and Charles Dyer. Toward human action recognition using dynamic perceptual organization. In *Looking at People Workshop*, Chambery, France, 1993. IJCAI.

[3] C. H. Anderson, P. J. Burt, and G. S. van der Wal. Change detection and tracking using pyramid transform techniques. In *Intelligent Robots and Computer Vision*, volume 579, Cambridge, MA, September 1985. SPIE.

[4] G. E. P. Box and G. M. Jenkins. *Time Series Analysis, Forecasting and Control*, chapter 6–7. Holden-Day, San Francisco, rev. ed. edition, 1976. 575 pp.

[5] N. Chiba, K. Muraoka, H. Takahashi, and M. Miura. Two-dimensional visual simulation of flames, smoke and the spread of fire. *Jrnl. of Visualization and Computer Animation*, 5:37–53, 1994.

[6] A. D. Cliff and J. K. Ord. Space-time modeling with an application to regional forecasting. *Trans. Inst. British Geographers*, 66:119–128, 1975.

[7] Ed. David S. Ebert. *Texture and Modeling — a Procedural Approach.* Academic Press, 1994. ISBN 0-12-228760-6.

[8] Kristine Gould and Mubarak Shah. The trajectory primal sketch: A multi-scale scheme for representing motion characteristics. In *Proc. CVPR '89*, pages 79–85, San Diego, CA, June 1989. IEEE Computer Society Press.

[9] Michael Grunkin. *On the Analysis of Image Data Using Simultaneous Interaction Models.* PhD thesis, Inst. of Mathematical Modeling and Operations Research, Technical University of Denmark, Lyngby, August 1993. no. 67.

[10] Michael Grunkin. Spectral texture statistics for classification. Working paper, 1994.

[11] Michael Grunkin. Personal communication, 1994–95.

[12] Michael Grunkin. Estimation and identification of causal SARMA models. Submitted to Pattern Recognition, 1995.

[13] Michael Grunkin, Per R. Andresen, and Knut Conradsen. The MORSE estimator for mixed noncausal SARMA models. 1994. Submitted to IEEE Transactions on Image Processing, 10-95.

[14] James D. Hamilton. *Time Series Analysis*. Princeton Univ. Press, 1994. ISBN 0-691-04289-6.

[15] Robert M. Haralick. Statistical and structural approaches to texture. *Proc. IEEE*, 67(5):786–804, May 1979.

[16] David Heeger. Optical flow from spatiotemporal filters. In *Proc. First Intl. Conf. Comp. Vision*, London, England, June 1987.

[17] David J. Heeger and Alex P. Pentland. Seeing structure through chaos. In *IEEE Workshop on Motion*, Charlestown, NC, 1986.

[18] R. L. Kashyap. Analysis and synthesis of image patterns by spatial interaction models. In L. Kanal and A. Rosenfeld, editors, *Progress in Pattern Recognition*, volume I. New York: North Holland, 1981.

[19] Rangasami Kashyap. Characterization and estimation of two-dimensional ARMA models. *IEEE T. Info. Theory*, IT-30(5):736–745, 1984.

[20] Rangasami L. Kashyap and Alireza Khotanzad. A model-based method for rotation invariant texture classification. *IEEE T. Patt. Analy. and Mach. Intell.*, PAMI-8(4):472–481, July 1986.

[21] T.J. Kung and W.A. Richards. Inferring "water" from images. In Whitman Richards, editor, *Natural Computation*, pages 224–233. MIT Press, 1988.

[22] Jae S. Lim. *Two-dimensional Signal and Image Processing*. Prentice Hall, 1990. ISBN 0-13-935322-4.

[23] Fang Liu and Rosalind W. Picard. Periodicity, directionality, and randomness: Wold features for perceptual pattern recognition. In *Proc. Int. Conf. Pat. Rec.*, volume II, pages 184–185, Jerusalem, Israel, October 1994. Also *MIT Media Lab Perceptual Computing TR* 293.

[24] Fang Liu and Rosalind W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. Perceptual Computing, Media Laboratory 320, MIT, Cambridge, MA, March 1995.

[25] Lennart Ljung. *System Identification: Theory for the User*. Prentice-Hall, 1987. ISBN 0-13-881640-9.

[26] John Makhoul. Linear prediction: A tutorial review. In *Proc. IEEE*, volume 63, pages 561–580, April 1975.

[27] Jianchang Mao and Anil K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.

[28] Makoto Miyahara and Yasuhiro Yoshida. Mathematical transform of (R, G, B) color data to Munsell (H, V, C) color data. *Visual Communications and Image Processing*, 1001:650–657, 1988. SPIE.

[29] Randal C. Nelson and Ramprasad Polana. Qualitative recognition of motion using temporal texture. *Comp. Vis., Graph., and Img. Proc.*, 56(1):78–89, July 1992.

[30] C. H. Perry and R. W. Picard. Synthesizing flames and their spreading. In *Proceedings of the Fifth Eurograhics Workshop on Animation and Simulation*, Oslo, Norway, September 1994. Also *MIT Media Lab Perceptual Computing TR* 287.

[31] Phillip E. Pfeifer and Stuart Jay Deutsch. Identification and interpretation of first order space time ARMA models. *Technometrics*, 22(3):397–408, August 1980.

[32] Phillip E. Pfeifer and Stuart Jay Deutsch. A three-stage iterative procedure for space-time modeling. *Technometrics*, 22(1):35–47, February 1980.

[33] R. W. Picard and T. P. Minka. Vision texture for annotation. *Journal of Multimedia Systems*, 3:3–14, 1995. Also *MIT Media Lab Perceptual Computing TR* 302.

[34] Ramprasad Polana. Detecting activities. In *Proceedings CVPR '93*, pages 2–7, 1993.

[35] Ramprasad Polana and Randal Nelson. Low level recognition of human motion. In *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 77–82, Austin, TX, 1994.

[36] Kris Popat and Rosalind W. Picard. Novel cluster probability models for texture synthesis, classification, and compression. In *Proc. SPIE Visual Communication and Image Proc.*, volume 2094, pages 756–768, Boston, Nov. 1993. Also *MIT Media Lab Perceptual Computing TR* 234.

[37] Lawrence Rabiner and Biing hwang Juang. *Fundamentals of Speech Recognition*. PTR Prentice-Hall, 1993. ISBN 0-13-015157-2.

[38] S. Seitz and R. Dyer. Detecting irregularites in cyclic motion. In *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 178–185, Austin, TX, 1994.

[39] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *Computer Graphics (SIGGRAPH) Proc.*, pages 369–375. ACM, 1993.

[40] Mihran Tuceryan and Anil K. Jain. Texture analysis. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *The Handbook of Pattern Recognition and Computer Vision*, chapter 2.1, pages 235–276. World Scientific Publishing Co., 1993.

[41] L. van Gool, P. Dewaele, and A. Oosterlinck. Texture analysis anno 1983. *Comp. Vis., Graph., and Img. Proc.*, pages 335–357, 1985.

[42] John Y. A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE T. Image Proc.*, 3(5):625–638, September 1994.

[43] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, Wokingham, England, 1992.

[44] Harry Wechsler. Texture analysis – a survey. *Signal Processing*, 2(3):271–282, July 1980.

[45] William W. S. Wei. *Time Series Analysis: Univariate and Multivariate Methods*. Addison-Wesley, 1990. ISBN 0-201-15911-2.

[46] P. Whittle. On stationary processes in the plane. *Biometrika*, 41:434–49, 1954.