SPATIAL AND TEMPORAL MODELS FOR TEXTURE-BASED VIDEO CODING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Fengqing Zhu

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2006

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

Upon the completion of this dissertation, I realized that there will never be the perfect words to express my gratitude towards those who have inspired, encouraged and motivated me through my years at Purdue. Yet, these names and faces will always remain deep in my heart.

First of all, I would like to thank my major advisor, Professor Edward J. Delp. I thank him for taking me under his guidance. He has provided me with the opportunity to expand my learning horizon by becoming a member of the VIPER lab. I am grateful for his constant encouragement of independent thinking and learning through struggles. Moreover, I have enjoyed many non-academic related conversations with him, from which I have grew further respect for his broad and diverse knowledge of various topics.

I would like to thank my other committee members, Professor Charles A. Bouman and Professor M. J. T. Smith, for their advice, guidance and criticism. I have enjoyed my learning experience in Professor Bouman's Image Processing classes and am grateful for his patience and insightful suggestions.

I would like to thank the Nokia Research Center for their sponsorship on the research in this dissertation. Their generous support has made this dissertation possible.

For the past year and a half I joined the VIPER lab, I have been lucky to be surrounded by my colleagues who are bright individuals capable of critical thinking. I appreciate the support and friendship of my fellow colleagues in the VIPER lab: Golnaz Abdollahian, Ying Chen, Nitin Khana, Dean King-Smith, Liang Liang, Limin Liu, Anthony Martone, Aravind Mikkilineni, Ka Ki Ng, and Carlos Wang.

I would like to especially thank Golnaz and Ka Ki for their great work on the temporal motion analysis of the texture-based video coding; to Limin, for her consul-

tation on H.264 related topics; to Aravind, for his extraordinary patience in helping me with software programming and other computer related issues.

I would like to thank Oriol Guitart, Dr. Hyung Cook Kim, Dr. Zhen Li, Ashok Mariappan, and Dr. Hwayoung Um who have recently graduated from Purdue. I have enjoyed working with them in the VIPER lab and wish them the best in the future.

I would like to thank past members of the VIPER lab for their inspiration and insight: Dr. Eugene T. Lin, Dr. Yuxin Liu, and Dr. Cuneyt Taskran.

Finally, I would like to thank my parents for their enduring support and encouragement. They have made many sacrifices throughout the years I have been away from home so that I can pursue my academic career. I thank them for giving me life and the opportunity to view the world with different perspectives.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Zhu, Fengqing M.S.E.C.E., Purdue University, December, 2006. Spatial and Temporal Models for Texture-Based Video Coding. Major Professor: Edward J. Delp.

In this dissertation, we investigate spatial and temporal models for texture analysis and synthesis. The goal is to use these models to increase the coding efficiency for video sequences containing textures. The models are used to segment texture regions in a frame at the encoder and synthesize the textures at the decoder. These methods can be incorporated into a conventional video coder (e.g. H.264) where the regions to be modeled by the textures are not coded in a usual manner but texture model parameters are sent to the decoder as side information. We showed that this approach can reduce the data rate by as much as 15%.

1. INTRODUCTION

In the past two decades, conventional hybrid video codecs have succeeded in increasing the video quality while reducing the data rate [1]. One way to increase the coding efficiency beyond the data rates achievable by modern codecs, such as H.264, is to not encode all the pixels in the sequence. In early coding systems this was achieved by either reducing the size of the frame and/or a combination of frame skipping. The quality of the reconstructed sequence was, of course, reduced.

In 1959, Schreiber and colleagues proposed a coding method he called, Synthetic Highs, which introduced the concept of dividing an image into textures and edges [2]. Two different approaches were then described to encode each type of structure in the image. This approach, used in image coding, was later extended by using a model of the Human Visual System (HVS) and a statistical model of the texture pixels in a frame [3–5]. The goal is to determine where "insignificant" texture regions in the frame are located and then use a texture model for the pixels in that region. By "insignificant" pixels we mean regions in the frame that the observer will not notice has been changed. The encoder then fits the model to the image and transmits the model parameters to the decoder which uses the model to reconstruct the pixels. An example of this approach is shown in Figure 1.1 where the black area in the frame on the right is not transmitted but reconstructed by the decoder from a texture region in a previous frame shown on the left. Since a frame is not homogeneous one may need to use different types of models for various texture regions in a frame. This leads to the need to first segment the frame into regions [6].

The problem with using this approach in video is that if each frame is encoded separately the areas that have been reconstructed with the texture models will be obvious when the video is displayed. This then requires that the texture to be modeled both spatially and temporally [7]. The mean square error (MSE) of a sequence that

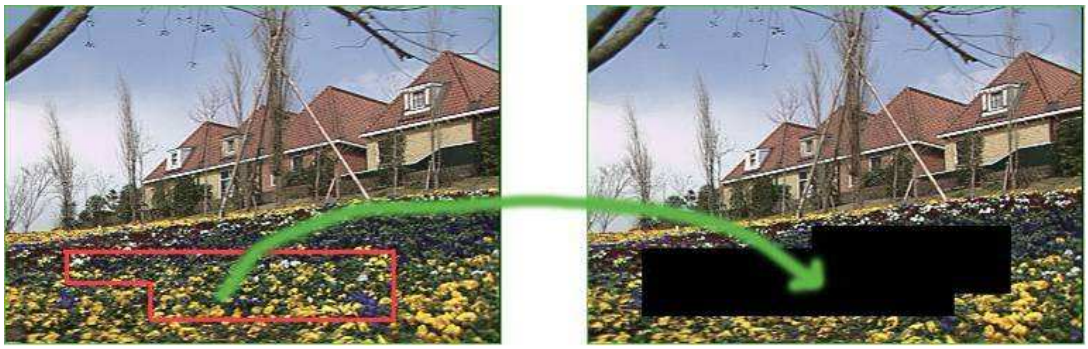


Fig. 1.1. General Approach: Texture and Motion Models Used to Construct a Frame.

has been encoded using this approach will be quite large when compared with a typical codec but when the sequence is displayed it may be very acceptable for many applications such as mobile video where the screen size is small.

In contrast to earlier standards, MPEG-4 [8] and H.264/AVC [9] allow one to encode a video sequence with frame skipping techniques in order to optimize the video coding process. We should note that skipping frames could lead to poor reconstruction of the video since this technique can overlook the content characteristics of the skipped frames. It is better to efficiently code video and exploit the content of the scene. Segmentation based methods (e.g. texture methods) are an example of this class. Recently, efficient content-based video coding approaches have been explored in which frames are segmented into various regions based on similarity constraints. Different coding schemes may used on the segmented regions in such a way that the overall data rate can be reduced while maintaining the visual quality. For example, the idea of dividing each frame into background/foreground regions based on temporal similarity and then using global motion estimation to reconstruct the background region is found in [10]. Another approach, reported by Wiegand and colleagues, is described in [11–14], where a coder was designed using the fact that textures such as grass, water, sand, and flowers can be synthesized with acceptable perceptual quality instead of coding them using conventional methods. Since the latter has a higher data rate in order to represent the details in the textures which are not visually important, the proposed approach can be used to increase the overall coding efficiency. The issues then are the trade-offs between data rate, modeling efficiency, and image quality. It is this approach that we investigated in this report.

A general scheme for video coding using texture analysis and synthesis is illustrated in Figure 1.2. The texture analyzer identifies homogeneous regions in a frame and labels them as textures. This step can be done using several texture segmentation strategies [5], for example, in [11–15] a feature-based method using the MPEG-7 color and edge descriptors for similarity estimation was examined. To ensure the temporal consistency of the identified textures throughout the video sequence, global motion

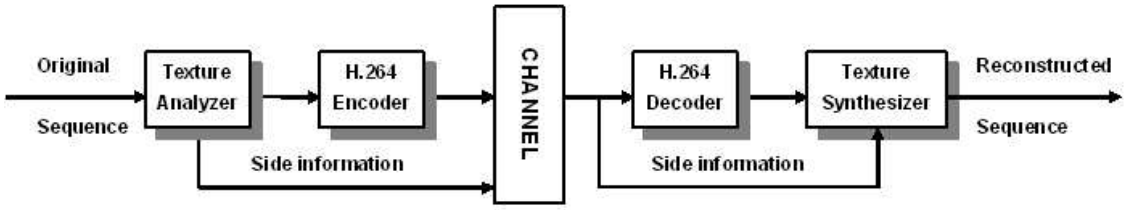


Fig. 1.2. Textured-Based Video Coding: System Overview.

models are used for these regions. The texture regions are warped from frame-to-frame using the motion model to ensure temporal consistency in the segmentation. The warping of a texture region is illustrated in Figure 1.1. A set of motion parameters for each texture region is sent to the texture synthesizer at the decoder as side information. The output of the texture analyzer is passed to a conventional video codec, e.g. H.264/AVC with synthesizable regions labeled as skip macroblocks. At the decoder, frames are partially reconstructed except for the synthesizable parts which are inserted later by the texture synthesizer using the side information.

In out research, we investigate spatial and temporal models for texture analysis and synthesis. We will show that excellent performance can be obtained using simpler models that have been proposed. We will also discuss the use of more innovative texture modeling methods and describe how they effect the performance of our coder. Our goal is to examine various texture modeling strategies and determine how well they can be used to model the texture regions and estimate what impact they will have on the data rate.

This dissertation is organized as follows. In Chapter 2 we concentrate on examining spatial texture models that are based on simple features and statistical models. In Chapter 3 we explore low complexity approaches to spatial-temporal motion models. In Chapter 4 we show the experimental results we obtained using our proposed method. In Chapter 5 we conclude our work and discuss potential research opportunities for this topic.

2. SPATIAL TEXTURE MODELS

2.1 Background

The first step in implementing the system described in Figure 1.2 is to extract regions in a frame that correspond to the same texture. Techniques for texture segmentation are used in applications which can be categorized into feature-based methods, model-based methods, spatial-frequency methods and structural methods [5,16]. In feature-based methods, characteristics of homogeneous regions are chosen as the texture features such as the co-occurrence matrix or geometrical features such as edges [17]. Model-based methods assume that the texture is described by a stochastic model and uses model parameters to segment the texture regions. Examples of such methods are found in [18] where a multiresolution Gaussian autoregressive model is described and in [3] where an image model is formulated using a seasonal autoregressive time series. Subband decomposition, especially the use of wavelets, is often seen in spatial-frequency methods [19]. Structural methods are based on the notion that textures are composed of primitives that are well-defined and spatially repetitive [5,20].

2.2 Texture Segmentation

Texture segmentation is often a two step process in which features are first obtained, followed by the segmentation step which is a "grouping" operation of the homogeneous regions based on the feature properties and a grouping metric [17,18].

2.2.1 Split and Merge

In [11, 12], a combination of quadtree segmentation and region growing was described using a split and merge method to generate coarse masks for the spatial texture regions [19]. A quadtree representation is used with a region splitting algorithm to recursively describe the texture from the root node, which represents the entire image or video frame, to the leaf nodes with each representing a coherent texture region.

Similar to this technique but far simpler, we split a frame into 16x16 non-overlapping macroblocks, with each macroblock acting as the root node for the quadtree. Each macroblock was decomposed into four non-overlapping regions. A macroblock is considered to be "classified" if its four regions have pairwise similar statistical properties. For similarity comparison, features based on the MPEG-7 descriptors for color and edges were examined, similar to the features used in [11,12], in addition we also examined other features. The resulting texture regions were "flagged" as candidates that were used by the motion models and these areas are not coded by the video encoder. We modified the algorithm described in [11,12] by using a much simpler version of the color features and not implementing the entire MPEG-7 color descriptor. We found that the edge features reported in [11] did not improve the quality of the segmented results for many sequences. We found that the edges obtained by using a simple edge operator such as the Kirsch operator performed very well [21]. This will be described in more detail later. In the merging step, a region growing algorithm is used to merge macroblocks labeled as "classified." The feature vector of a "classified" macroblock is compared with its four neighbors to determine whether or not they can be merged using the similarity criterion.

2.2.2 Thresholds and Metrics

If the distance between the feature vectors of two blocks is less than a threshold, then two blocks are considered to be similar. The threshold is a proportion of the maximum possible distance between two feature vectors which depends both on the

metric selection and the descriptor choice. A threshold of one means that two blocks are always similar since one is the maximum possible distance, while a threshold of zero means that two blocks are similar only if they are identical. The optimal threshold was experimentally determined by testing several key frames of a video sequence and then used on all frames of the video.

The two metrics chosen for the distance between two feature vectors are the ℓ_1 norm and the Earth Mover's distance (EMD). In the following definitions, H_a and H_b denote two histograms with B bins, and

$$C_1 = \sqrt{\frac{N_{H_b}}{N_{H_a}}}, \quad C_2 = \frac{1}{C_1}, \quad N_{H_b} = \sum_{j=1}^{B} H_a(j), \quad N_{H_a} = \sum_{j=1}^{B} H_b(j).$$

The ℓ_1 norm is the absolute bin-wise difference between histogram H_a and H_b.

$$d_{L_1}(H_a, H_b) = \sum_{j=1}^{B} |C_1 \cdot H_a(j) - C_2 \cdot H_b(j)| \tag{2.1}$$

d_{L_1} is normalized by $2 \cdot \sqrt{N_{H_a} \cdot N_{H_b}}$ to the range of $[0, 1]$.

The Earth Mover's distance [22, 23] is the minimum cost that must be paid to transform one distribution into another. If we define the bin population of the first of the two histograms as hills and the corresponding population of the second histogram as valleys, then the EMD represents the minimum "earth" transportation cost from the hills to the valleys. The greater the distance between the provider (histogram #1) and the receiver bin (histogram #2), the higher the transportation costs. Histograms with different locations of most of the "earth" concentration will be labeled as very different, while histograms with similar shapes and noise deviations will be seen as similar. EMD is more robust than histogram matching techniques, in that it emphasizes comparing the shapes of the histograms and is less affected by noise, shifting and scaling. This also makes EMD eligible for compensating for lighting variations when used in combination with the color feature.

Computing the Earth Mover's distance is based on a solution to the transportation problem from linear optimization, known as the Monge-Kantorovich problem [23]. This can be formalized as the following linear programming problem: let $P =$

$\{(p_1, w_{p1}), \ldots (p_m, w_{pm})\}$ be the first signature with m clusters, where p_i is the cluster representative and w_{pi} is the weight of the cluster, $Q = \{(q_1, w_{q1}), \ldots (q_m, w_{qm})\}$ the second signature with n clusters; and $D = [d_{ij}]$ the ground distance matrix where d_{ij} is the ground distance between clusters p_i and q_i.

We want to find a flow $F = [f_{ij}]$, where f_{ij} is the flow between p_i and q_i, that minimizes the overall cost

$$WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \tag{2.2}$$

subject to the following constraints:

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n \tag{2.3}$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{pi} \qquad 1 \leq i \leq m \tag{2.4}$$

$$\sum_{i=1}^{n} f_{ij} \leq w_{qi} \qquad 1 \leq j \leq n \tag{2.5}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min \left(\sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{n} w_{qi} \right) \tag{2.6}$$

Constraint (2.3) allows moving "supplies" from P to Q and not vice versa. Constraint (2.4) limits the amount of supplies that can be sent by the clusters in P to their weights. Constraint (2.5) limits the clusters in Q to receive no more supplies than their weights; and constraint (2.6) forces to move the maximum amount of supplies possible. We call this amount the *total flow*. Once the transportation problem is solved, and we have found the optimal flow F, the earth mover's distance is defined in equation (2.7) as the work normalized by the total flow:

$$EMD(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{2.7}$$

2.3 Features and Models

In our work we have concentrated on examining spatial texture models that are based on simple features and statistical models. Our goal was to examine the work

reported in [11] and determine the effect of other types of texture models. In addition to simpler models, we wanted to then extend the work using more sophisticated models including Gibbs textures [17] and unsupervised models that we have previously reported in [18].

2.3.1 Color Feature

The color feature we examined is not the MPEG-7 color feature used in [11, 12]. We used a simpler version of the feature that omits the use of the Haar transform. Our color feature is basically a color histogram defined in the Hue-Saturation-Value color space with fixed color space quantization. The HSV model defines a color space in terms with three constituent components. Hue ranges from $0 - 360^\circ$ and specifies color type such as yellow, red or blue. Saturation specifies the "vibrancy" or "purity" of the color and has a range between 0 and 1. Value specifies the brightness of the color and also ranges from 0 to 1. The equivalent partitioning of the HSV color space can differ from 16 to 256 bins. In our application, we used the highest resolution to achieve the best possible results given this descriptor. The above setting consists of 16 hue levels, with each level corresponding to four saturation levels per luminance value. This yields 64 bins per luminance value. Since four luminance values are used in the SCD, this leads to total of 256 bins. If H_{vs}^{h} represents a color with a quantized hue at a quantized saturation and quantized value, then the colors in the reference histogram are sorted in the following order: $H_{00}^{0} \cdots H_{03}^{0} H_{10}^{0} \cdots H_{13}^{0} H_{20}^{0} \cdots H_{23}^{0} H_{30}^{0} \cdots H_{33}^{0} \cdots H_{33}^{1} \cdots H_{33}^{15}$

2.3.2 Edge Detection

MPEG-7 Edge Descriptor

The MPEG-7 "Edge Histogram Descriptor" used in [11, 12] represents local edge distribution by categorizing edges into five types: vertical, horizontal, 45 degree di-

agonal, 135 degree diagonal and nondirectional edges, as in Figure 2.1. The image is divided into 16 equal sized, non-overlapping subimages. The frequency of occurrence of each edge type is determined for each subimage, generating total of 80 (16×5) histogram bins. To localize edge distribution to a certain area of the image, we di-

Fig. 2.1. Edge Classes Used in MPEG-7 Feature.

vide the image space into 4x4 sub-images as shown in Figure 2.2. Then, for each sub-image, we generate an edge histogram to represent edge distribution in the sub-image. To define different edge types, the sub-image is further divided into small square blocks called image-blocks. Independent of the size of the image, our exper-

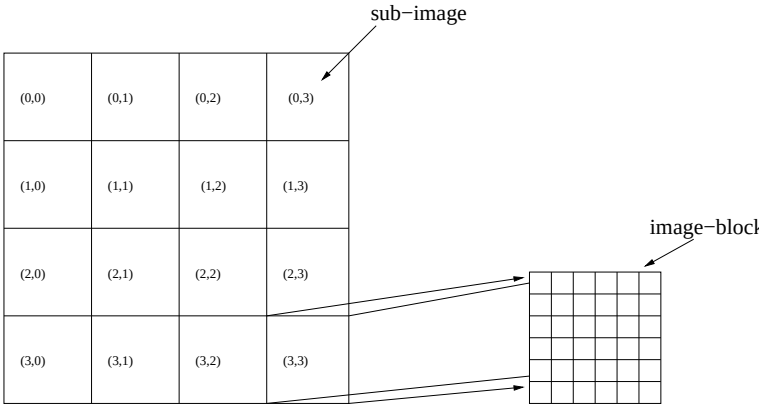
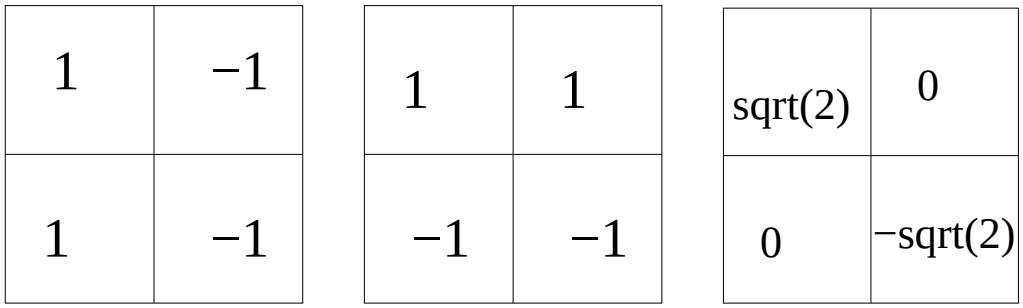


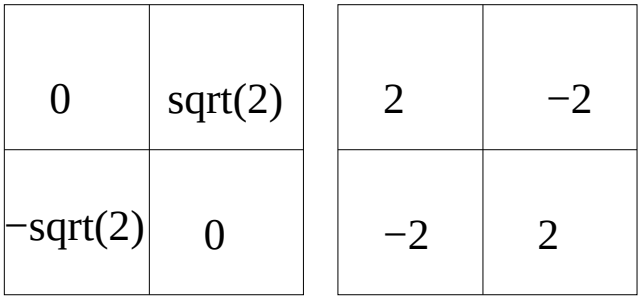
Fig. 2.2. A Sub-Image and Image-Block.

iments showed that a predetermined number of image-blocks of approximately 1100 are desirable for capturing good directional edge features. Edge features are extracted from the image-block as shown in Figure 2.2 where the image-block is further divided into four sub-blocks. The luminance mean values for the four sub-blocks are used

for edge detection. More specifically, the mean values of the four sub-blocks are obtained and convolved with the filter coefficients shown in Figure 2.3 to obtain the edge magnitudes. From the five directional edge strengths, if the maximum is greater than a threshold (Th_{edge}), then we select that the block has the corresponding edge type. More specifically, label the sub-blocks from 0 to 3 as in Figure 2.4. For the k^{th} ($k = 0, 1, 2, 3$) sub-block of the $(i, j)^{th}$ image-block, we can obtain the average gray level $A_k(i, j)$.



(a) ver_edge_filter (b) hor_edge_filter (c) dia45_edge_filter



(d) dia135_edge_filter (e) nond_edge_filter

Fig. 2.3. Filters Used for Edge Detection.

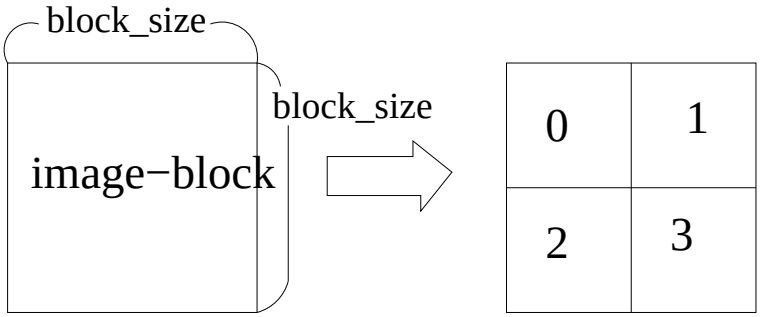


Fig. 2.4. Sub-blocks and Their Labeling.

Adopting the same labeling order as in Figure 2.4, we have the coefficients of the vertical edge filter in Figure 2.3(a) as follows:

$$\begin{aligned} ver\_edge\_filter(0) &= 1 \\ ver\_edge\_filter(1) &= -1 \\ ver\_edge\_filter(2) &= 1 \\ ver\_edge\_filter(3) &= -1 \end{aligned}$$

Similarly, we can represent the filter coefficients for other edge filters as shown in Figure 2.3(b), 2.3(c), 2.3(d), 2.3(e). Using five edge filters in Figure 2.3, we can obtain five edge strengths for the image block (i, j) using equation (2.8) - (2.12) as follows.

$$ver\_edge\_stg(i,j) = \sum_{k=0}^{3} |A_k(i,j) \times ver\_edge\_filter(k)| \quad (2.8)$$

$$hor\_edge\_stg(i,j) = \sum_{k=0}^{3} |A_k(i,j) \times hor\_edge\_filter(k)| \quad (2.9)$$

$$dia45\_edge\_stg(i,j) = \sum_{k=0}^{3} |A_k(i,j) \times dia45\_edge\_filter(k)| \quad (2.10)$$

$$dia135\_edge\_stg(i,j) = \sum_{k=0}^{3} |A_k(i,j) \times dia135\_edge\_filter(k)| \quad (2.11)$$

$$nond\_edge\_stg(i,j) = \sum_{k=0}^{3} |A_k(i,j) \times nond\_edge\_filter(k)| \quad (2.12)$$

If the maximum value among five edge strengths obtained from the above equations is greater than the threshold (Th_{edge}), then the image-block is considered to have the corresponding edge in it. For our application, the image is divided into 16x16 macroblocks, each consists of 5 bins representing the frequency of occurrence of the five types of edges.

The Kirsch Edge Operator

The MPEG-7 "Edge Histogram Descriptor" described above did not provide good edge features for texture segmentation, thus a better edge operator was examined. The Kirsch edge operator [21, 24] is a nonlinear edge detector, which estimates the gradient direction. The edge gradient is estimated as one of 8 discrete directions starting at 0° and using steps of 45°. An analytic description for the Kirsch operator can be expressed as:

$$h_{nm} = max_{z=1,\cdots,8} \sum_{i=-1}^{1} \sum_{j=-1}^{1} g_{ij}^{(z)} \cdot f_{n+i,m+j} \tag{2.13}$$

with direction templates:

$$g^{(1)} = \begin{bmatrix} +5 & +5 & +5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, g^{(2)} = \begin{bmatrix} +5 & +5 & -3 \\ +5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, g^{(3)} = \begin{bmatrix} +5 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & -3 & -3 \end{bmatrix},$$

$$g^{(4)} = \begin{bmatrix} -3 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & +5 & -3 \end{bmatrix}, g^{(5)} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ +5 & +5 & +5 \end{bmatrix}, g^{(6)} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & +5 \\ -3 & +5 & +5 \end{bmatrix},$$

$$g^{(7)} = \begin{bmatrix} -3 & -3 & +5 \\ -3 & 0 & +5 \\ -3 & -3 & +5 \end{bmatrix}, g^{(8)} = \begin{bmatrix} -3 & +5 & +5 \\ -3 & 0 & +5 \\ -3 & -3 & -3 \end{bmatrix}. \tag{2.14}$$

The 8-kernel Kirsch filters are used on the luminance pixels of each 16x16 macroblock. We then find the strongest edge gradient strength for each pixel in a mac-

roblock. Finally, we generate histogram accumulations for each macroblock with 8 bins corresponding to each type of edge strength.

2.3.3 Gabor Filter

2D Gabor filters describe various features related to the local power spectrum of a signal or image. These features correspond to different spatial variations in the image. Motivated by the studies of human perception, Gabor filters have been proposed in the literature as for many texture-segmentation applications [25–27].

A Gabor filter has as its impulse response a Gaussian envelope modulated by a sinusoidal plane wave along the x-axis:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)+2\pi jWx\right] \tag{2.15}$$

where W is the modulation frequency, x, y are coordinates in the spatial domain, and σ_x and σ_y are the standard deviations in the x and y direction. The 2D Fourier Transform is:

$$G(u,v) = exp\left\{-\frac{1}{2}\left[\frac{(u-W)^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}\right]\right\} \tag{2.16}$$

where $\sigma_u = \frac{1}{2\pi\sigma_x}$, $\sigma_u = \frac{1}{2\pi\sigma_y}$, and u and v are coordinates in the frequency domain. A Gabor filter-bank consists of Gabor filters with Gaussians of different sizes modulated by sinusoidal plane waves of different orientations from the same mother Gabor filter as defined in equation (2.15):

$$g_{m,n}(x,y) = a^{-m} g(\tilde{x},\tilde{y}), \quad a>1 \tag{2.17}$$

where $\tilde{x} = a^{-m}(x\cos\theta + y\sin\theta)$, $\tilde{y} = a^{-m}(-x\sin\theta + y\cos\theta)$, $\theta = n\pi/K$ (K = total orientation, and $n = 0, 1, ..., K-1$), and $g(\cdot,\cdot)$ is defined in equation (2.15).

Given an image $I_E(r,c)$ of size $H \times W$, the discrete Gabor filtered output is given by a 2D convolution, which is usually implemented in the frequency domain simply as the product:

$$I_{g_{m,n}}(r,c) = \sum_s \sum_t I_E(r-s, c-t) {g_{m,n}}^*(s,t),\ m = 0,1,...,S-1, n = 0,...,K-1 \tag{2.18}$$

where * indicates the complex conjugate. We can obtain the mean and standard deviation of the energy of the filtered image, which are often used as Gabor features.

$$\mu_{mn} = \frac{\sum_r \sum_c |I_{g_{m,n}}(r,c)|}{H \times W} \tag{2.19}$$

$$\sigma_{mn} = \frac{\sqrt{\sum_r \sum_c (|I_{g_{m,n}}(r,c)| - \mu_{mn})^2}}{H \times W} \tag{2.20}$$

Then we have a feature vector of size $S \times K$:

$$f = [\mu_{00} \quad \sigma_{00} \quad \cdots \quad \mu_{(S-1)(K-1)} \quad \sigma_{(S-1)(K-1)}]$$

The following design described in [25] guarantees the adjacent half-peak contours touch each other, after choosing the number of orientations K, the number of scales S, and the upper and lower center frequencies U_h and U_l:

$$a = \left(\frac{U_h}{U_l}\right)^{\frac{1}{S-1}}, \ \sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\ln 2}}$$
$$\sigma_v = \tan\left(\frac{\pi}{2K}\right)\sqrt{\frac{U_h^2}{2\ln 2} - \sigma_u^2}, \ W = U_h$$
$$m = 0, 1, \cdots, S-1, \quad n = 0, 1, \cdots, K-1$$

Based on this design, we propose to extract the Gabor features using the following. In our work, the highest and lowest frequencies of the Gabor filter-bank will be experimentally chosen to suit our desired texture analysis. We will select 6 orientations and 3 scales for our Gabor filter-bank, i.e. 18 Gabor filters. The following parameters will be used as the initial conditions in our work: $S = 3$, $K = 6$, $U_h = 0.75$, and $U_l = 0.125$. We will obtain the mean and standard deviation of the energy of each Gabor filtered image as features. Therefore, there will be 36 Gabor features extracted from each 16x16 macroblock:

$$f = [\mu_{00} \quad \sigma_{00} \quad \cdots \quad \mu_{25} \quad \sigma_{25}]$$

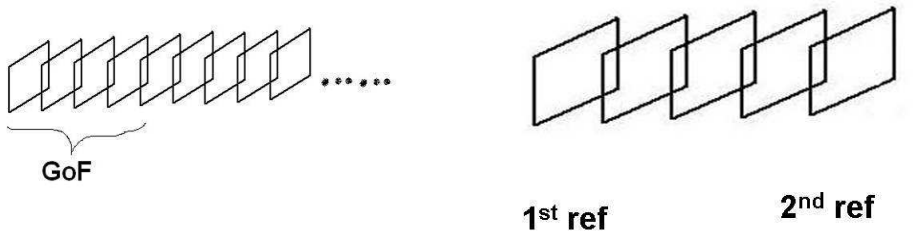
It should be noted that we did not extensively test the use of Gabor features in our experiments due to time constraints. We feel that these features should be examined in future work.

3. TEMPORAL TEXTURE MODELS

3.1 Background

The spatial texture models described in the previous sections operate on each frame of a given sequence independently of the other frames of the same sequence. This yields inconsistency in the segmentation across the sequence and can be very noticeable when the video is viewed. One can address this problem by using spatial-temporal texture models [7] or using something similar to motion compensation for the texture models in each frame [11].

The former approach exploits HVS models but requires relatively complicated motion estimation methods to fit the model to the sequence. We explored low complexity approaches to spatial temporal models. The latter methods use a texture catalog to keep track of the synthesizable textures identified in previous frames similar to that reported by Wiegard in [12]. The texture catalog contains information about the textures present in the sequence. The texture catalog is initialized with the feature vectors of the texture region identified in a starting frame or the first frame in which at least one texture is present. The textures identified in the following frames are compared to the existing textures in the catalog. When new textures are identified, the catalog is updated with corresponding feature vectors. In this report we have examined a modification of the Wiegand approach [12] with a consistency check using the motion model described below.



(a) GOF (b) Key Frames

Fig. 3.1. Illustration of Group of Frames(GOF). (a) A sequences of GOFs, (b) Two key frames within a GOF.

3.2 Temporal Analysis

3.2.1 Warping

In order to maintain temporal consistency of the texture regions, the video sequence is first divided into groups of frames (GoF). Each GoF consists of two key frames (the first and last frame of the GoF) and several middle frames to be modeled with textures between the two key frames. This is illustrated in Figure 3.1.

The key frames will either be I or P frames when they are coded. For every texture region in each of the middle frames we look for similar textures in both key frames. The corresponding region (if it can be found in at least one of the key frames) is then mapped into the segmented texture region. There are three possible cases, the texture is only found in the first key frame, the last key frame, or it is found in both key frames. This is illustrated in Figure 3.2. In most cases, similar textures can be found in both key frames. In this case, the the texture that results in the smallest error will be considered. The details of the metrics for the error will be described later in this section.

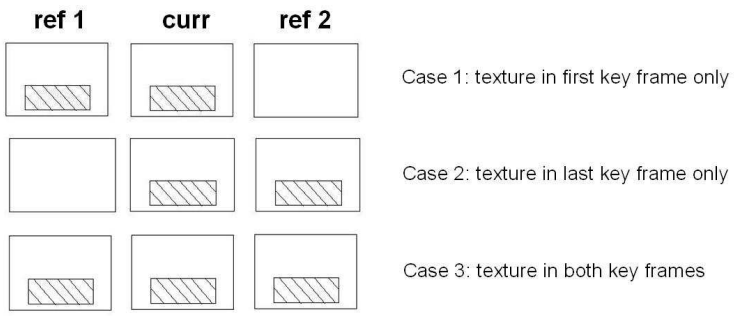


Fig. 3.2. Choice of Key Frame.

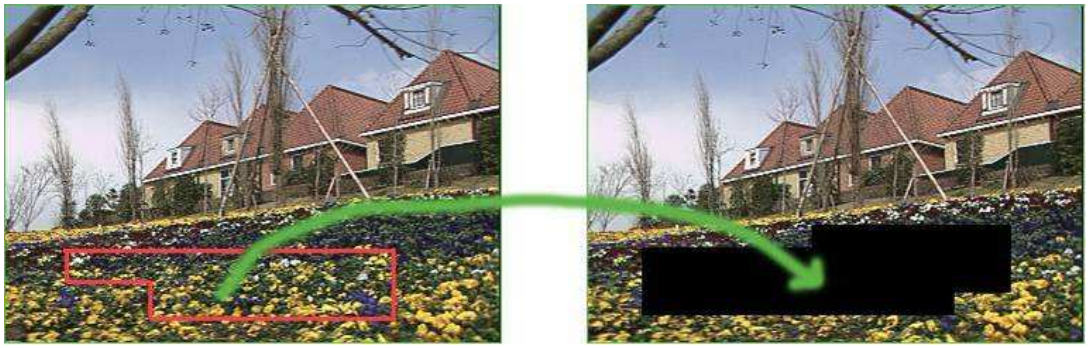


Fig. 3.3. The Motion Model is Used For Warping the Texture From The Key Frame to Texture Frame.

3.2.2 Motion Model

The texture regions are warped from frame-to-frame using a motion model to provide temporal consistency in the segmentation as illustrated in Fig.3.3. The mapping is based on a global motion assumption for every texture region in the frame i.e. the displacement of the entire region can be described by just one set motion parameters. We modified a 8-parameter (i.e. planar perspective) motion model to compensate the global motion [28]. This can be expressed as:

$$\begin{aligned} x' &= \frac{a_1 + a_3 x + a_4 y}{1 + a_7 x + a_8 y} \\ y' &= \frac{a_2 + a_5 x + a_6 y}{1 + a_7 x + a_8 y} \end{aligned} \tag{3.1}$$

Where (x, y) is the location of the pixel in the texture frame and (x', y') is the corresponding mapped coordinates. The planar perspective model is suitable to describe arbitrary rigid object motion if the camera operation is restricted to rotation and zoom. It is also suitable for arbitrary camera operation, if the objects with rigid motion are restricted planar motion. In practice these assumptions often hold over a short period of a GoF.

When an identified texture region in one of the texture frames is warped towards the key frame of the GoF, only the pixels of the warped texture region that lie within the corresponding texture region of the key frame of the GoF are used for synthesis.

Although this reduces the texture region in the texture frame, it is more conservative and usually gives better results.

The motion parameters $(a_0, a_1, \ldots, a_8)$ are estimated using a simplified implementation of a robust M-estimator for global motion estimation [28]. The weighting function is simplified to a rectangular function, i.e. a point is either weighted fully or considered as outliers and are discarded:

$$w(\epsilon) = \begin{cases} 1 & \epsilon^2 < c\mu \\ 0 & \epsilon^2 > c\mu \end{cases} \tag{3.2}$$

where ϵ, the residual, is obtained from the difference of the luminance between the actual and the warped pixels. The sum of squares of all the residuals except the outliers is minimized. Then μ is the average sum of squares of all N points within region R,

$$\mu = \frac{1}{N} \sum_{p \in R} \epsilon^2 \tag{3.3}$$

c is used to adjust the sensitivity of the optimization. With an increasing value of c, more pixels in both structure and unstructured areas are used for the iteration.

3.2.3 Parameter Optimization

We estimated the set of parameters using an iterative Gauss-Newton method. The process of warping and optimization is done for both key frames, hence two sets of motion parameters are estimated (each set corresponds to a key frame) and the set that has smallest MSE between the synthesized and original texture region is used We used as a "stopping criterion" the first minimum obtained in the cost function μ.

Unstructured image regions can have a bad impact on the motion parameter estimates results. The residuals will be small in unstructured areas even if the motion parameters estimates are poor. However, areas such as edges are likely to result in large areas even if the estimation is good. To ensure the adequate performance of the

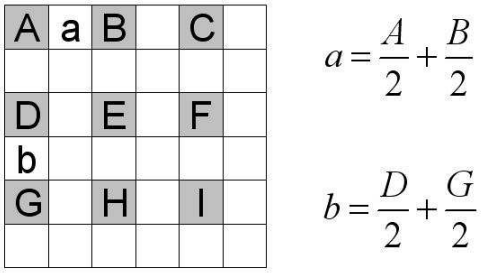


Fig. 3.4. Bilinear Interpolation.

estimator, the unstructured regions are detected and excluded from the estimation. The pixels in the unstructured regions are obtained by:

$$V = \begin{cases} 1 & |I_x| > d \cap |I_y| > d \\ 0 & \text{else} \end{cases} \tag{3.4}$$

In some cases a better estimate of the motion parameters may be obtained by first interpolating pixels and then doing the motion estimation [9]. The interpolation method is the same as the H.264 sub-pixel interpolator. We used bilinear interpolation for half pixels for the Y samples on all frames for our motion model. An example of the interpolated Y samples is shown in Figure 3.4. The set of motion parameters along with the flag to indicate the corresponding key frame are sent to decoder as side information.

3.3 Texture Synthesis

At the decoder, key frames and non-synthesizable parts of other frames are conventionally decoded. The remaining parts labeled as synthesizable regions are skipped by the encoder and their values remain blank. The texture synthesizer is then used to reconstruct the corresponding missing pixels.

With the assumption that the frame to frame motion can be described using a planar perspective model, then given the motion parameter set and the control

parameter that indicated which frame (first or last frame of the GoF) is used as the key frame, the texture regions can be reconstructed by warping the texture from the key frame towards each synthesizable texture region identified by the texture analyzer.

4. EXPERIMENTAL RESULTS

4.1 System Integration

The spatial-temporal models described in the previous sections were integrated into the H.264/AVC JM 10.2 reference software. I and P frames are conventionally coded; only B frames are candidates for texture synthesis. In the case where a B frame contains identified synthesizable texture regions, the corresponding segmentation mask, motion parameters as well as the control flags have to be transmitted as side information to the decoder. All macroblocks belonging to a synthesizable texture region are handled as skipped macroblocks in the H.264/AVC reference software. That is, all parameters and variables needed for decoding the macroblocks inside the slice (in decoding order) are set as specified for skipped macroblocks. This includes the reconstructed YUV samples that are needed for intra prediction, the motion vectors and the reference indices that are used for motion vector prediction. After all macroblocks of a frame are completely decoded, the texture synthesis is performed in which macroblocks belonging to a synthesizable texture region are replaced.

4.2 Results

The integrated video codec was tested using CIF sequences such as Flowergarden, Coastguard, Tabletennis, MissAmerica and Football. The following parameters were used for the H.264/AVC codec: QP=24, 28, 32, 36, 40 and 44; 3 B frames; 1 reference frame; CABAC; rate distortion optimization; no interlace; 30Hz frame frequency.

Figure 4.1 and Figure 4.2 show the results obtained for frame #58 and frame #90 of the Flowergarden sequence. The regions that our texture segmentation algorithm labeled as “insignificant” texture are indicated as the white region in the mask. Note

that system indicates that the flowers can be modeled by the texture system. These pixels would then not be coded by the conventional coder. Note that important parts of the frame would be encoded with high fidelity. This is different than the shape coder used in MPEG-4. These figures also show how the textures are generated in each frame using the motion model. To reduce the blockiness effect along the edges, we used a low pass filter on the edge pixels. A visual artifact may be created if the intensity of the light in the environment changes over time. To fix this problem, we transmit the difference between the average intensity of the texture region in the texture frame and the key frame as additional side information. At the synthesizer this will be added to the intensity of the synthesized pixels. Similar results for other test sequences are shown in Figures 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, and 4.10.

4.3 Estimating The Data Rate

The data rate savings for each test sequence is calculated by subtracting from the original data rate (coded with the H.264 test coder) the data rate savings for macroblocks that are not coded using the H.264 coder. The data rate used to construct the side information is then added to obtain the data rate for the texture coded video. The side information contains the coarse texture masks (macroblock-accurate), 8 motion parameters and 1 control flag to indicate which key frame is used as the reference frame (the first frame or the last frame of the GOF). The data rate for side information is 256 bits for the motion parameters and one bit for the control flag. The data rate for texture mask depends size of the texture mask. and is typically about 600 bits. Hence the side information is less than 1Kb per frame. The data rate can be described by the following:

- Average bit allocation per MB for a B-frame = total number of bits for B-frames (kb) $\div$ (number of B-frames $\times$ number of MBs per frame)
- Data rate savings for total skip MBs (kb) = average number of bits per MB for B-frame $\times$ number of skip MBs

- Total data rate for the sequence (using texture models) @ 30 Hz (kb/s) = original data rate @ 30 Hz (kb/s) - (data rate savings for total skip MBs (kb) - data allocation for total side information (kb)) ÷ number of total frames × 30 Hz

The maximum data rate savings for each of the test sequences occur when the quantization parameter is set to its smallest value within our range (QP=24). Table 4.1 shows the data rate savings obtained for the Flowergarden sequences for each quantization level. The visual quality was comparable to the quality of the decoded sequences using the H.264 codec. Notice that some of the data savings become negative when the data rate is below 300 kb/s for the compressed sequence, this is due to the fact that number of bits used for the side information is fix for all quantization levels. Results obtained for other test sequences are presented in Table 4.2, Table 4.3, Table 4.4, Table 4.5.

For some sequences, e.g. Missamerica and Tabletennis, it is possible to observe segmentation errors. This is caused by the the texture masks not properly matching from one frame to the next. In the Missamerica sequence this is caused by the fact that the subject's hair is very close to the same color as the background. In the Tabletennis sequence this is caused by a scene change and errors generated by the fast moving ball.

Table 4.1
Data Rate Savings Obtained for the Flowergarden Sequence.

Quantization Level	H.264 data rate [kb/s]	Texture data rate [kb/s]	Data Rate Savings
24	4564.75	3786.94	17.03%
28	2959.80	2502.32	15.46%
32	1698.88	1484.86	12.60%
36	864.66	797.47	7.77%
40	427.89	423.42	1.04%
44	226.68	238.88	-5.38%

Table 4.2
Data Rate Savings Obtained for the Coastguard Sequence.

Quantization Level	H.264 data rate [kb/s]	Texture data rate [kb/s]	Data Rate Savings
24	2221.62	1925.78	13.32%
28	1224.79	1093.56	10.71%
32	571.47	542.29	5.11%
36	255.35	268.28	-5.06%
40	123.42	147.59	-19.58%
44	61.96	89.51	-44.46%

Table 4.3
Data Rate Savings Obtained for the Football Sequence.

Quantization Level	H.264 data rate [kb/s]	Texture data rate [kb/s]	Data Rate Savings
24	3514.16	3031.24	13.74%
28	2122.53	1857.89	12.47%
32	1234.96	1098.95	11.01%
36	686.25	627.12	8.62%
40	373.63	356.61	4.56%
44	195.26	201.31	-3.10%

Table 4.4
Data Rate Savings Obtained for the Missamerica Sequence.

Quantization Level	H.264 data rate [kb/s]	Texture data rate [kb/s]	Data Rate Savings
24	395.88	370.20	6.49%
28	170.88	182.53	-6.82%
32	94.03	115.09	-22.40%
36	57.69	82.92	-43.74%
40	38.92	66.03	-69.65%
44	29.72	59.42	-93.21%

Table 4.5
Data Rate Savings Obtained for the Tabletennis Sequence.

Quantization Level	H.264 data rate [kb/s]	Texture data rate [kb/s]	Data Rate Savings
24	1139.12	1003.79	11.88%
28	662.51	606.73	8.42%
32	369.68	358.04	3.15%
36	210.45	219.93	-4.50%
40	127.54	146.21	-14.64%
44	82.61	105.35	-27.53%

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.1. Example of spatial and temporal model for the Flowergarden sequence (a) Frame 58 used as the original frame, (b) Frame 58 with mask for texture region and (c) Reconstructed Frame 58 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.2. Example of spatial and temporal model for the Flowergarden sequence (a) Frame 90 used as the original frame, (b) Frame 90 with mask for texture region and (c) Reconstructed Frame 90 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.3. Example of spatial and temporal model for the Coastguard sequence (a) Frame 3 used as the original frame, (b) Frame 3 with mask for texture region and (c) Reconstructed Frame 3 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.4. Example of spatial and temporal model for the Coastguard sequence (a) Frame 102 used as the original frame, (b) Frame 102 with mask for texture region and (c) Reconstructed Frame 102 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.5. Example of spatial and temporal model for the Football sequence (a) Frame 32 used as the original frame, (b) Frame 32 with mask for texture region and (c) Reconstructed Frame 32 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.6. Example of spatial and temporal model for the Football sequence (a) Frame 62 used as the original frame, (b) Frame 62 with mask for texture region and (c) Reconstructed Frame 62 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.7. Example of spatial and temporal model for the Missamerica sequence (a) Frame 52 used as the original frame, (b) Frame 52 with mask for texture region and (c) Reconstructed Frame 52 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.8. Example of spatial and temporal model for the Missamerica sequence (a) Frame 93 used as the original frame, (b) Frame 93 with mask for texture region and (c) Reconstructed Frame 93 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.9. Example of spatial and temporal model for the Tabletennis sequence (a) Frame 3 used as the original frame, (b) Frame 3 with mask for texture region and (c) Reconstructed Frame 3 with synthesized texture.

(a) Original Frame

(b) Texture Mask for Original Frame

(c) Reconstructed Frame

Fig. 4.10. Example of spatial and temporal model for the Tabletennis sequence (a) Frame 47 used as the original frame, (b) Frame 47 with mask for texture region and (c) Reconstructed Frame 47 with synthesized texture.

5. CONCLUSION AND FUTURE WORK

In this project, we investigated spatial and temporal models for texture analysis and synthesis. The goal was to use these models to increase the coding efficiency for video sequences containing textures. Models were used to segment texture regions in a frame at the encoder and synthesize the textures at the decoder. These methods were incorporated into a conventional video coder (e.g. H.264) where the regions modeled by the textures were not coded in a usual manner but texture model parameters were sent to the decoder as side information. We showed that this approach reduced the data rate by as much as 15%. The emphasis on our work was to examine the work reported by Wiegand and colleagues [11–14]. We also extended the approach by accomplishing the following:

1. developed a simpler version of the color texture feature that did not use the Haar Transform
2. developed a simpler and better edge feature using the Kirsch operator
3. investigated the use of the Earth Mover's distance as a texture metric
4. used a simpler motion model and convergence method and also used sub-pixel motion
5. used smoothing and texture boundary side information to reduce texture boundaries in decoded frames

The results look very promising but we feel more work needs to be done. We feel following topics need to further investigated:

- texture features - other types of texture features could used including Gabor methods, Co-Occurrence and Unsupervised approaches, and other types of content descriptors.

- motion models - the motion model we used was simple and we feel that this needs further study

- texture/motion side information - exactly what types of side information does the decoder need needs more work. We did not optimize this in our study.

- experimental results - many more sequences need to be investigated including SD and perhaps HD sequences at various data rates.

- segmentation errors and boundary reconstruction problems - these need further work to develop methods to detect when they will occur and the decoder can fix them.

LIST OF REFERENCES

LIST OF REFERENCES

[1] G. Sullivan and T. Wiegand, "Video compression - from concepts to the h.264/avc standard," in *Proceedings of the IEEE*, vol. 93, pp. 18–31, January 2005.

[2] W. F. Schreiber, C. F. Knapp, and N. D. Kay, "Synthetic highs, an experimental tv bandwidth reduction system," *Journal of Society of Motion Picture and Television Engineers*, vol. 68, pp. 525–537, August 1959.

[3] E. J. Delp, R. L. Kashyap, and O. Mitchell, "Image data compression using autoregressive time series models," *Pattern Recognition*, vol. 11, pp. 313–323, June 1979.

[4] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-generation image-coding techniques," *Proceedings of the IEEE*, vol. 73, pp. 549 – 574, April 1985.

[5] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, pp. 786–804, May 1979.

[6] H. A. Peterson, *Image Segmentation Using Human Visual System Properties with Applications in Image Compression.* PhD thesis, Purdue University, May 1990.

[7] C.-H. Peh and L.-F. Cheong, "Synergizing spatial and temporal texture," *IEEE Transactions on Image Processing*, vol. 11, pp. 1179 – 1191, October 2002.

[8] "Coding of audiovisual objects - part 2: Visual," 1999.

[9] "Text of committee draft of joint video specification," 2002.

[10] S. Liu, Z. Yan, J. W. Kim, and C. C. J. Kuo, "Global/local motion-compensated frame interpolation for low bitrate video," in *Proc. International Society of Optical Engineering*, (San Jose, CA), January 2000.

[11] P. Ndjiki-Nya, C. Stuber, and T. Wiegand, "Improved video coding through texture analysis and synthesis," in *Proceedings of the 5th International Workshop on Image Analysis for Multimedia Interactive Services*, (Lisboa, Portugal), April 2004.

[12] P. Ndjiki-Nya, O. Novychny, and T. Wiegand, "Video content an analysis using mpeg-7 descriptors," in *Proceedings of the First European Conference on Visual Media Production*, (London, Great Britain), March 2004.

[13] P. Ndjiki-Nya, A. S. B. Makai, H. Schwarz, and T. Wiegand, "Improved h.264/avc coding using texture analysis and synthesis," in *Proceedings of ICIP, IEEE International Conference on Image Processing*, (Barcelona, Spain), September 2003.

[14] P. Ndjiki-Nya, B. Makai, A. Smolic, H. Schwarz, and T. Wiegand, "Video coding using texture analysis and synthesis," in *Proceedings of PCS 2003, Picture Coding Symposium*, (St. Malo, France), April 2003.

[15] P. Ndjiki-Nya, O. Novychny, and T. Wiegand, "Merging mpeg-7 descriptors for image content analysis," in *Proceedings of ICASSP, IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Montreal, Canada), May 2004.

[16] K. I. Chang, K. W. Bowyer, and M. Sivagurunath, "Evaluation of texture segmentation algorithms," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, (Fort Collins, CO.), pp. 294–299, June 1999.

[17] I. Elfadel and R. Picard, "Gibbs random fields, cooccurrences, and texture modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 24 – 37, January 1994.

[18] M. L. Comer and E. J. Delp, "Segmentation of textured images using a multiresolution gaussian autoregressive model," *IEEE Transactions on Image Processing*, vol. 8, pp. 408 – 420, March 1999.

[19] J. R. Smith and S.-F. Chang, "Quad-tree segmentation for texture-based image query," in *accpeted for ACM Multimedia*, (San Francisco, CA.), October 1994.

[20] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Reading, MA: Addison-Wesley, 1992.

[21] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers in Biomedical Research*, vol. 4, pp. 315–328, June 1971.

[22] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Sixth International Conference on Computer Vision (ICCV'98)*, (Bombay, India), January 1998.

[23] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," *Technical Report STAN-CS-TN-98-86-Computer Science Department-Stanford University*, September 1998.

[24] L. O. Michael Seul and M. J. Sammon, *Practical Algorithms for Image Analysis with CD-ROM: Description, Examples, and Code*. Cambridge, United Kingdom: Cambridge University Press, 2000.

[25] B. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 18, pp. 837–842, August 1996.

[26] S. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on gabor filters," *IEEE Transactions on Image Processing*, vol. 11, pp. 1160–1167, October 2002.

[27] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Transactions on Image Processing*, vol. 11, pp. 467–476, April 2002.

[28] A. Smolic and J. Ohm, "Robust global motion estimation using a simplified m-estimator approach," in *Proceedings of the IEEE International Conference on Image Processing*, (Vancouver, Canada), September 2000.