Spatial and Temporal Models for Texture-Based Video Coding

Fengqing Zhu, Ka Ki Ng, Golnaz Abdollahian and Edward J. Delp

Video and Image Processing Lab (VIPER)
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907, USA

ABSTRACT

In this paper, 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%.

Keywords: video coding, texture modeling

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 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 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. An example of such approach, reported by Wiegand and colleagues, is described in [8, 9], 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 MSE. 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

This work was sponsored by a grant from Nokia. Address all correspondence to E. J. Delp, ace@ecn.purdue.edu.
increase the overall coding efficiency. The issues then are the trade-offs between data rate, modeling efficiency, and image quality.

A general scheme for video coding using texture analysis and synthesis is illustrated in Figure 2. The texture analyzer identifies homogeneous regions in a frame and labels them as textures. This step can be done by using several texture segmentation strategies [5], for example, in [8–10] a feature-based method using the MPEG-7 color and edge descriptors for similarity estimation is examined. To ensure the temporal consistency of the identified textures throughout the video sequence, global motion 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. A set of 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 this paper, 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.

2. TEXTURE ANALYSIS

2.1. Spatial Texture Models

The first step in implementing the system described in Figure 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 [5,11]. 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 [12]. 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 [13] 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 [14]. Structural methods are based on the notion that textures are composed of primitives that are well-defined and spatially repetitive [5, 15].

2.1.1. Texture Segmentation

Texture segmentation 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 [12, 13]. In [8, 9], 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 [14]. 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 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 color and edges were examined, similar to the MPEF-7 descriptors used in [8, 9]. 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 [8, 9] 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 [8] 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 [16]. 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.

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 ℓ1 norm \(d_{\ell_1}\) and the Earth Mover’s distance (EMD) [17, 18]. 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 \(\ell_1\) norm is the absolute bin-wise difference between histogram \(H_a\) and \(H_b\).

\[
d_{\ell_1}(H_a, H_b) = \sum_{j=1}^{B} |C_1 \cdot H_a(j) - C_2 \cdot H_b(j)|
\]

\(d_{\ell_1}\) is normalized by \(2 \cdot \sqrt{N_{H_a} \cdot N_{H_b}}\) to the range of \([0,1]\). The Earth Mover’s distance [17, 18] 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.

2.1.2. 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 [8] 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 [12] and unsupervised models that we have previously reported in [13].

The color feature we examined is not the MPEG-7 color feature used in [8,9]. 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 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 color feature, this leads to total of 256 bins. If $H_{hsv}$ 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: $H^0_{00} \cdots H^0_{03} \cdots H^0_{010} \cdots H^0_{013} \cdots H^0_{020} \cdots H^0_{023} \cdots H^0_{030} \cdots H^0_{033} \cdots H^1_{33} \cdots H^{15}_{33}$

The MPEG-7 “Edge Histogram Descriptor” used in [8,9] represents local edge distribution by categorizing edges into five types: vertical, horizontal, 45 degree diagonal, 135 degree diagonal and nondirectional edges, as in Figure 3. 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 ($16 \times 5$) histogram bins.

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 [16,19] is a nonlinear edge detector, which estimates the gradient direction. The edge gradient is estimated as one of 8 discrete directions starting at $0^\circ$ and using steps of $45^\circ$. An analytic description for the Kirsch operator can be expressed as:

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

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

2.2. Temporal Analysis

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 [8]. 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.

Figure 3. Edge Classes Used in MPEG-7 Feature.
similar to that reported by Wiegard in [9]. 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 [9] with a consistency check using the motion model described below.

2.2.1. Temporal Texture Models

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 reference frames (first and last frame of the considered GoF) and several middle frames between the two reference frames. The reference 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 reference frames. The corresponding region (if it can be found in at least one of the reference frames) is then mapped into the segmented texture region. There are three possible cases, the texture is only found in the first reference frame, the last reference frame, or it is found in both reference frames. In most cases, similar textures can be found in both reference 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.

The texture regions are warped from frame-to-frame using a motion model to provide temporal consistency in the segmentation as illustrated in Fig.4. 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 [20]. This can be expressed as:

\[
x' = \frac{a_1 + a_3x + a_4y}{1 + a_7x + a_8y}
\]

\[
y' = \frac{a_2 + a_5x + a_6y}{1 + a_7x + a_8y}
\]  

Where \((x, y)\) is the location of the pixel in the current 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 middle frames (current frame) is warped towards the reference frame of the GoF, only the pixels of the warped texture region that lie within the corresponding texture region of the reference frame of the GoF are used for synthesis. Although this reduces the texture region in the current 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 [20]. The
weighting function is simplified to a rectangular function, i.e. a point is either weighted fully or considered as outliers and are discarded:

\[
    w(\epsilon) = \begin{cases} 
        1 & \epsilon^2 < c\mu \\
        0 & \epsilon^2 > c\mu 
    \end{cases} \quad (4)
\]

where \(\epsilon\), 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 \(\mu\) is the average sum of squares of all \(N\) points within region \(R\),

\[
    \mu = \frac{1}{N} \sum_{p \in R} \epsilon^2 \quad (5)
\]

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.

We estimated the set of parameters using an iterative Gauss-Newton method. The process of warping and optimization is done for both reference frames, hence two sets of motion parameters are estimated (each set corresponds to a reference 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 \(\mu\). Unstructured image regions can have a negative 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 estimator, the unstructured regions are detected and excluded from the estimation. The pixels in the unstructured regions are obtained by:

\[
    V = \begin{cases} 
        1 & (|I_x| > d) \cap (|I_y| > d) \\
        0 & \text{else} 
    \end{cases} \quad (6)
\]

In some cases a better estimate of the motion parameters may be obtained by first interpolating pixels and then doing the motion estimation [21]. 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 5. The set of motion parameters along with the flag to indicate the corresponding reference frame are sent to decoder as side information.

2.2.2. Texture Synthesis

At the decoder, reference 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 reference frame, the texture regions can be reconstructed by warping the texture from the reference frame towards each synthesizable texture region identified by the texture analyzer.
3. EXPERIMENTAL RESULTS

3.1. 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.

3.2. 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 6 shows the results obtained for frame #58 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 current frame and the reference frame as additional side information. At the synthesizer this will be added to the intensity of the synthesized pixels. Similar results for the Coastguard sequence is shown in 7.

3.3. Data Rate Estimation

The data rate savings for each test sequence is estimated 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 the 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 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 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. Result obtained for the Coastguard sequence is presented in Table 2.

4. CONCLUSIONS AND FUTURE WORK

In this paper, 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

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

<table>
<thead>
<tr>
<th>Quantization Level</th>
<th>H.264 data rate [kb/s]</th>
<th>Texture data rate [kb/s]</th>
<th>Data rate Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>2221.62</td>
<td>1925.78</td>
<td>13.32%</td>
</tr>
<tr>
<td>28</td>
<td>1224.79</td>
<td>1093.56</td>
<td>10.71%</td>
</tr>
<tr>
<td>32</td>
<td>571.47</td>
<td>542.29</td>
<td>5.11%</td>
</tr>
<tr>
<td>36</td>
<td>255.35</td>
<td>268.28</td>
<td>-5.06%</td>
</tr>
<tr>
<td>40</td>
<td>123.42</td>
<td>147.59</td>
<td>-19.58%</td>
</tr>
<tr>
<td>44</td>
<td>61.96</td>
<td>89.51</td>
<td>-44.46%</td>
</tr>
</tbody>
</table>
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 results look very promising but we feel more work needs to be done including the examination of other
texture features (e.g. Gabor methods) and further work needs to be done on the motion model. Some of the
sequences suffered from segmentation errors that caused poor image quality on playback. These types of errors
need to be detected before the frame are used for texture synthesis.

REFERENCES
   of the IEEE, vol. 73, no. 4, pp. 549 – 574, April 1985.
5. R. M. Haralick, “Statistical and structural approaches to texture,” Proceedings of the IEEE, vol. 67, no. 5,
6. H. A. Peterson, “Image segmentation using human visual system properties with applications in image
   compression,” Ph.D. dissertation, Purdue University, May 1990.
   Proceedings of the 5th International Workshop on Image Analysis for Multimedia Interactive Services,
   Lisboa, Portugal, April 2004.
9. P. Ndjiki-Nya, O. Novychay, and T. Wiegand, “Video content an analysis using mpeg-7 descriptors,” Pro-
10. ———, “Merging mpeg-7 descriptors for image content analysis,” Proceedings of ICASSP, IEEE International
    ceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, Fort Collins, CO.,
13. M. L. Comer and E. J. Delp, “Segmentation of textured images using a multiresolution gaussian autore-
17. Y. Rubner, C. Tomasi, and L. J. Guibas, “A metric for distributions with applications to image databases,”
    Sixth International Conference on Computer Vision (ICCV’98), Bombay, India, January 1998.
18. Y. Rubner, C. Tomasi, and L. Guibas, “The earth mover’s distance as a metric for image retrieval,” Technical
19. L. O. Michael Seul and M. J. Sammon, Practical Algorithms for Image Analysis with CD-ROM: Description,