SPATIAL TEXTURE MODELS FOR VIDEO COMPRESSION

Marc Bosch, Fengqing Zhu 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 integrate several spatial texture tools into a texture-based video coding scheme. We implemented texture techniques and segmentation strategies in order to detect texture regions in video sequences. These textures are analyzed using temporal motion techniques and are labeled as skipped areas that are not encoded. After the decoding process, frame reconstruction is performed by inserting the skipped texture areas into the decoded frames. We are able to show an improvement over previous texture-based implementations in terms of compression efficiency.

Index Terms— Video coding, image segmentation, texture modeling, feature extraction

1. INTRODUCTION

New applications such as real time mobile video communications and movies-on-demand delivered to portable receivers are driving an increasing interest in novel techniques for increasing the coding efficiency of video compression methods. One way to increase the coding efficiency beyond the data rates achievable by modern codecs, such as H.264/MPEG-4 AVC, is to not encode all the pixels in the sequence. 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 [1]. 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 and a statistical model of the texture pixels in a frame [2, 3]. The goal is to determine where "insignificant" texture regions or "detail-irrelevant" regions in the frame are located and then use a texture model for the pixels in the region. By "insignificant" pixels we mean regions in the frame that the observer will not notice what has been changed. The encoder then fits the model to the image and transmits the model parameters to the decoder as side information which uses the model to reconstruct the pixels.

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. An example of such approach is described in [4], where a video coder was designed using the notion that textures such as grass, water, sand, and flowers can be synthesized with acceptable perceptual quality instead of coding them using mean square error. Since the latter has a higher data rate in order to represent the details in the textures which are not visually important, the 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.

This work was partially supported by grants from the Indiana 21^{st} Century Research and Technology Fund and by Nokia. Address all correspondence to E. J. Delp, ace@ecn.purdue.edu

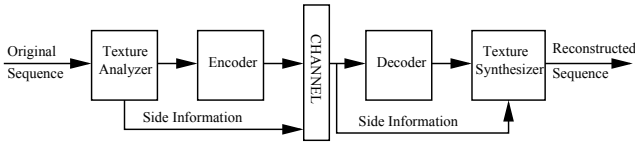


Fig. 1. Texture Coding System Overview.

Fig. 2. Model Used to Reconstruct Texture Regions.

A general scheme for video coding using texture analysis and synthesis is illustrated in Figure 1. The texture analyzer identifies homogeneous regions in a frame and labels them as textures. To ensure the temporal consistency of the identified textures throughout the video sequence, global motion models are used to warp texture regions from frame-to-frame. 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, 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 texture side information as shown in Figure 2 [5].

In this paper, we examine various configurations for the texture analyzer by means of several texture models. We experimentally evaluate the performance of these methods with respect to coding efficiency (data rate) and visual quality.

2. TEXTURE ANALYSIS

2.1. Overview

In texture analysis, there are two major issues that need to be addressed, namely texture feature extraction and texture boundary detection or segmentation. Feature extraction is used to measure local texture properties in an image, followed by boundary detection or segmentation to group the features into similar regions.

Typically, four approaches have been used for texture feature extraction: structural methods, statistical or feature-based methods, model-based methods and transform or spatial-frequency methods. Structural methods represent textures by well-defined primitives (microtextures) and spatial arrangements (macrotextures) of the primitives [6]. Statistical methods represent textures by non-deterministic properties that model the relationships of the gray levels of the image. Particularly, second-order statistical methods have been developed that achieve higher performance than structural techniques [7]. The most popular second order statistical features are derived from the co-occurrence matrix [8]. Model-based texture analysis, using fractal and stochastic models [9, 10], describe an image texture by the use of repeated geometric and stochastic models. Transform methods, such as Fourier, Gabor and wavelet transforms, measure local texture properties in the frequency domain.

Once the texture features are extracted and feature vectors are formed, texture segmentation then groups regions in an image that have similar texture properties. Based on [11], several texture analysis algorithms were investigated in this paper and separately integrated into the texture analyzer of our video compression scheme to compare data rate savings. The following texture features + segmentation schemes have been considered in our experiments: Gray level Co-occurrence matrix + Split and Merge, Gray level Co-occurrence matrix + K-means algorithm, and Gabor filter + K-means algorithm.

2.2. Texture Features

In previous work [12, 4, 5], color and edge based features were examined. In this paper, we introduce more sophisticated features including Gray Level Co-occurrence Matrix and Gabor filters. We will show in Section 3 that these methods increase the performance of the texture-based video coding approach without quality loss.

One aspect of texture is concerned with the spatial distribution among the gray levels of the pixels in a local area. Haralick $et\ al.$ introduced the Gray Level Co-occurrence Matrix (GLCM) in [8], suggesting an approach to characterize the two-dimensional spatial dependence of the pixels in an image. GLCM describes the spatial relationship between pixels by means of the occurrence of pairs of gray levels in the image. The GLCM is an estimate of the two dimensional probability distribution of the pixels and depends on the the distance between the pixels and their angular displacement [8]. Once the GLCM is generated, features, such as entropy, uniformity, and contrast are obtained and formed into feature vectors used for texture segmentation. In our simulations we have considered four angular relationships: 0°, 45°, 90° and 135°. Five distance values or spatial resolutions have been tested: 1, 2, 3, 5 and 10. We have used the following features: uniformity, dissimilarity, correlation, entropy and contrast. The features are described in more detail in [8].

Gabor filters describe properties related to the local power spectrum of a signal and have been used for texture features [13]. A Gabor impulse response in the spatial domain consists of a sinusoidal plane wave of some orientation and frequency, modulated by a two-dimensional Gaussian envelope and is given by:

$$h(x,y) = exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)\right]\cos\left(2\pi Ux+\varphi\right) \qquad (1)$$

In our work, we used a Gabor filter-bank proposed in [14]. It is highly suitable for our use where the texture features are obtained by subjecting each image (or in our case each macroblock) to a Gabor filtering operation in a window around each pixel and compute the mean and the standard deviation of the energy of the filtered image. A Gabor filter-bank consists of Gabor filters with Gaussians of several sizes modulated by sinusoidal plane waves of different orientations from the same Gabor-root filter as defined in Equation (1), it can be represented as:

$$g_{m,n}(x,y) = a^{-m}h(\tilde{x},\tilde{y}), \quad a>1 \qquad (2)$$

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, $n = 0, 1, ..., K-1$, and $m = 0, 1, ..., S-1$), and $h(\cdot,\cdot)$ is defined in Equation (1). Given a video frame $I_E(r,c)$ of size $H \times W$, the discrete Gabor filtered output is given by a 2D convolution:

$$I_{g_{m,n}}(r,c) = \sum_{s,t} I_E(r-s, c-t){g_{m,n}}^*(s,t), \qquad (3)$$

As a result of this convolution, the energy of the filtered image is computed and then, the mean and standard deviation are estimated and used as features. In our implementation, the following parameters are considered: 4 scales (S=4), and 8 orientations (K=8).

2.3. Texture Segmentation

In our schemes, two segmentation strategies are used by the texture analyzer: Split and Merge algorithm [15], and K-means clustering [16].

Split and Merge algorithm uses the entire feature image as the input of the first stage and successively splits the image into subregions based on the homogeneity of the features of each sub-region. Following this operation, the merge process is performed by comparing the distance between the feature vector of each sub-region and its four neighbors with a predetermined threshold to decide whether or not they can be combined [15].

Clustering methods label parts of the image by partitioning the feature vectors of the image into compact, well-defined and separated clusters. The non-hierarchical method we used selects an initial partition with K clusters [16]. Next, a new partition is generated by examining each component in the population and assigning it to one of the clusters depending on a metric. Then new cluster centers are obtained as the centroids of the cluster. The last two steps are repeated until the centroids are fixed. The metrics considered in our experiments are Euclidean distance and Manhattan distance.

2.4. Temporal Analysis

The spatial texture models referred in the previous section operate on each frame of a given sequence independently of the other frames of the same sequence. This may yield an inconsistency in segmentation across the sequence. To maintain temporal consistency of the texture regions, they are warped from frame-to-frame using a motion model[5]. 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.The motion parameters are estimated using a simplified implementation of a robust M-estimator for global motion estimation [5] and sent as side information to the synthesizer.

3. EXPERIMENTAL RESULTS

3.1. System Integration

The texture models described in the previous sections were integrated into the H.264/AVC JM 11.0 reference software. In our im-

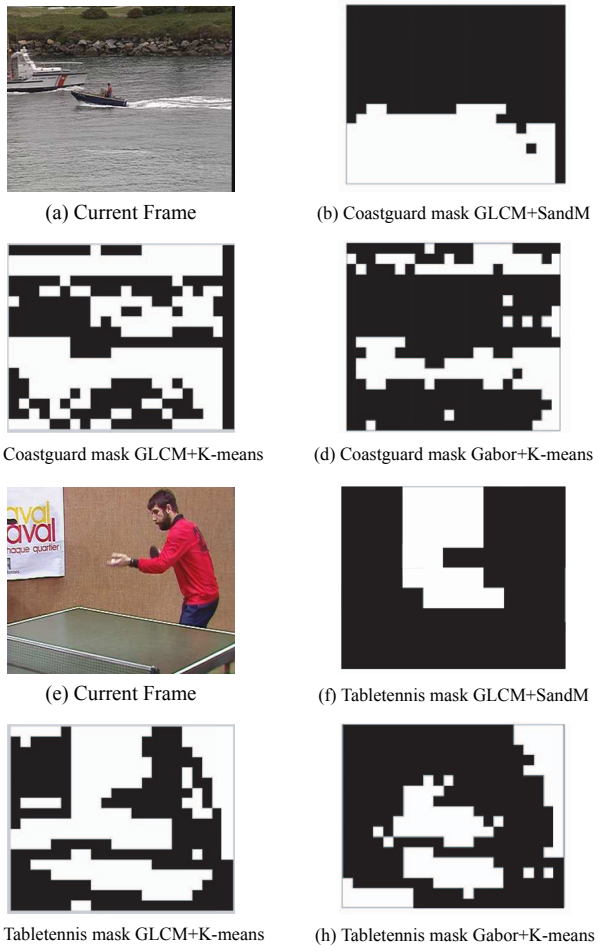
plementation, the video sequence was first divided into groups of frames (GoF). Each GoF consisted of two reference frames (first and last frame of the considered GoF) and several middle frames between the two reference frames. The reference frames were conventionally coded as I or P frames; the middle frames were encoded as B frames that were candidates for texture synthesis [5]. For every texture region in each of the middle frames, the texture analyzer looked for similar textures in both reference frames. The corresponding area (if it can be found in at least one of the reference frames) was then mapped into the segmented texture region based on a global motion model. When a B frame contained identified synthesizable texture regions, the corresponding segmentation masks, motion parameters as well as the control flag to indicate which reference frame was used were transmitted as side information to the decoder [5]. All macroblocks belonging to a synthesizable texture region were handled as skipped macroblocks in the H.264/AVC reference software. Hence, all parameters and variables used for decoding the macroblocks inside the slice, in decoder order, were set as specified for skipped macroblocks. After all macroblocks of a current frame were completely decoded, texture synthesis was performed in which macroblocks belonging to a synthesizable texture region were replaced with the textures identified in the corresponding reference frame.

Our goal was to measure the performance of the models we tested in terms of data rate savings. By data rate savings we mean, the absolute difference between the original data rate i.e. the entire video sequence coded with the H.264 codec and the texture data rate. The texture data rate is obtained by adding the data rate of the video sequences without the irrelevant-detail parts that are not coded, to the side information which is no more than 1.25Kb per frame [5]. The side information contains the coarse texture masks (typically about 800 bits), 8 motion parameters (256 bits) which ensure temporal consistency at the decoder and 1 control flag (1 bit) to indicate which frame is used as the reference frame (the first or the last frame of the GOF).

We encoded several CIF video sequences such as *coastguard, tabletennis*, and *flowergarden*, as well as some high definition video sequences (*waterfall* and *race*) using our proposed methods. All testing sequences contained large texture and homogeneous areas useful to demonstrate the phenomenon that with a similar representation of the textures the observer would not notice any difference in terms of visual quality. In our experiments, we used the following parameters for the H.264/AVC codec: Quantization Parameter for intra and inter frames = 24, 32 and 44; 1 reference frame; 3 B frames; CABAC; rate distortion optimization; no interlace; constant channel; 30 frames per second.

3.2. Segmentation Results

Figure 3 shows that the Gray Level Co-occurrence matrix and the Split and Merge algorithms provide a good segmented image, detecting a considerable number of possible skipped macroblocks (16x16 pixels) so that the data rate can be reduced. GLCM + K-means was able to identify more than one type of texture and homogeneous regions for each video frame. For example, Figure 3(g) shows that we identified the wall, the table, the poster, the net and the red t-shirt of the tabletennis sequence each as a unique candidate for the detail-irrelevant area. The second solution (GLCM + K-means), and particularly, the K-means algorithm had an advantage in comparison to other methods, since it detected several diverse texture areas and labeled them each as a unique texture region. Finally, the third scheme using Gabor filter + K-means clustering did not provide better results than GLCM + K-means algorithm because it detected less unique texture areas than GLCM + K-means (see Figure 3). This was caused by the fact that the features were not as homogeneous as the GLCM features. Gabor + K-means did provide a better solution than previous methods in terms of computation time due to the fact that the number of computational operations in Gabor feature was less than the GLCM algorithm. This is a significant advantage, especially for long video sequences or high definition video.



(a) Current Frame (b) Coastguard mask GLCM+SandM
(c) Coastguard mask GLCM+K-means (d) Coastguard mask Gabor+K-means
(e) Current Frame (f) Tabletennis mask GLCM+SandM
(g) Tabletennis mask GLCM+K-means (h) Tabletennis mask Gabor+K-means

Fig. 3. Original frames for *coastguard* and *tabletennis* sequences and their texture masks for all the proposed schemes. The white areas represent the detail-irrelevant texture regions to be synthesized.

3.3. Coding Performance

The main contribution of this paper lies in the introduction of spatial texture models for use in video coding. We have presented three schemes that we integrated into the analyzer texture block diagram in Figure 1. From our experiments we have observed the following:

1. The K-means segmentation algorithm identified more textures as synthesizable areas. Although, depending on the features used, K-means could also identify irrelevant regions as texture areas. The Split and Merge algorithm only recognized one texture area usually the largest one.
2. In terms of data rate savings, the Gray Level Co-occurrence

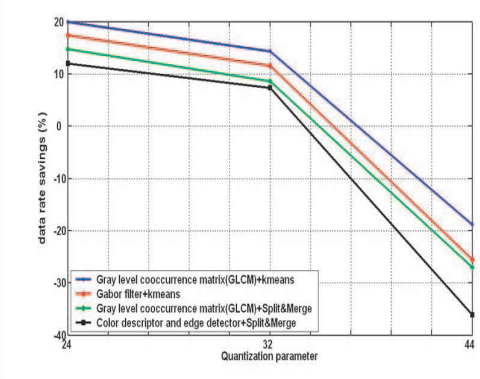


Fig. 4. Data rate savings as a function of the H.264 encoder quantization parameter. Our three schemes compared to the results reported in [5]. Each value is the average over all sequences tested (flowergarden, coastguard, tabletennis, waterfall and race).

matrix + K-means algorithm performed the best as shown in Figure 4, where each scheme was tested and averaged for the CIF and HD video sequences. For *coastguard* sequence we measured data rate savings of up to 25.91% using GLCM + K-means, 22.16% using Gabor + K-means and 16.02% using GLCM + Split and Merge, all with quantization parameter = 24. This approach provided better data rate savings than we previously reported in [5].

3. When computational time was considered, there was no remarkable variation between segmentation algorithms. However, there was a notable distinction between the features. The Gabor filter was faster than Gray level Co-occurrence matrix when the same texture segmentation algorithm was used.

4. The side information did not influence the data rate savings in most scenarios. Only when the quantization parameter is larger than 36 did the side information have an effect (see Figure 4). This was due to the fact that the number of bits required by the side information remained constant over the diverse quantization parameters. Since the data rate of the side information and the B frames with skipped macroblocks was larger than the data rate required by the reference video codec to encode the original video sequence in these cases, the overall data rate savings for texture-based video coding became negative.

In this paper, we have investigated spatial models for texture analysis. The goal was to use these models to increase the coding efficiency for video sequences containing textures and homogeneous areas. We showed that this approach reduced the data rate by as much as 26%. In terms of visual quality, all three schemes (GLCM + S&M, GLCM + K-means, Gabor + K-means) indicated similar results. They were comparable to the visual quality obatined by using the H.264 codec.

4. REFERENCES

[1] 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.

[2] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-generation image-coding techniques," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 549–574, April 1985.

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

[5] F. Zhu, K. Ng, G. Abdollahian, and E. J. Delp, "Spatial and temporal models for texture-based video coding," *Proceedings of the Visual Communications and Image Processing Conference*, San Jose, California, January 2007.

[6] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, May 1979.

[7] J. Weszka, C. Deya, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *Proceedings of the IEEE International Conference on Image Processing*, Atlanta, Georgia, October 8-12 2006.

[8] R. Haralick, K. Shanmugan, and I. Dinstein, "Textural feature for image classification," *IEEE Transactions on Systems, man and cybernetics*, vol. SMC-3, no. 6, pp. 610–621, November 1973.

[9] A. Pentlard, "Fractal-based description of natural sciences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 661–674, December 1976.

[10] M. Strzelecki and A. Materka, "Markov random fields as models of textured biomedical images," *Proceedings of the 20th National Conference on Circuit theory and electronic networks*, Kolobrzeg, Poland, September 1997, pp. 493–498.

[11] K. Chang, K. Bowyer, and M. Sivagurunath, "Evaluation of texture segmentation algorithms," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Fort Collins, Colorado, June 1999, pp. 249–299.

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

[13] P. Kruizinga, N. Petkov, and S. E. Grigorescu, "Comparison of texture features based on gabor filters," *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, Washington, DC, USA, September 1999, p. 142.

[14] A. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Los Angeles, CA, November 1990.

[15] J. R. Smith and S.-F. Chang, "Quad-tree segmentation for texture-based image query," *Proceedings of the second ACM international conference on Multimedia*, San Francisco, CA., October 1994.

[16] V. Ramos and F. Muge, "Image colour segmentation by genetic algorithms," *ArXiv Computer Science e-prints*, December 2004.