VIDEO CODING USING MOTION CLASSIFICATION

Marc Bosch, Fengqing Zhu and Edward J. Delp

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

ABSTRACT
In this paper we present a video coding approach similar to texture-based methods but based on motion models. We consider motion perception properties instead of spatial texture properties of the video sequence. We integrate a motion classification algorithm to separate foreground objects containing noticeable motion from the background. These background areas are labeled as skipped areas that are not encoded. After decoding, frame reconstruction is performed by inserting the skipped background into the decoded frames. We are able to show as much as 15% an improvement over previous texture-based implementations in terms of video compression efficiency.

Index Terms—video coding, motion analysis, coding efficiency.

1. INTRODUCTION
One way to increase the coding efficiency achievable by modern video codecs is to not encode all the pixels in the video sequence. In particular, regions belonging to areas that the viewer may not perceive can be skipped or coded using different approaches. In 1959, Synthetic Highs was proposed [1] which introduced the concept of dividing an image into textures and edges. Two approaches were 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 [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 these regions. By “insignificant” pixels we mean regions in a frame that the observer will not notice what has been changed. The encoder 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. An example of texture based methods is described in [4, 5, 6], where coding methods are described using the concept that textures such as grass, water, sand, and flowers can be synthesized with acceptable perceptual quality instead of coding them using more classical approaches.

We propose in this paper an approach, similar to texture-based coding, by examining the motion properties of a video sequence. We present a method consisting of “not encoding” areas in a video frame that are “secondary” based on a simple model of human motion perception in order to explore the trade-offs between data rate, modeling efficiency, and image quality. By “not encoding” regions we mean that, similar to texture based methods, motion model parameters will be sent to the decoder as side information and the decoder will use this information to locally reconstruct the frame. Our motion analysis and synthesis video coding scheme is illustrated in Figure 1. We will also compare the performance of both texture based and motion based schemes in terms of coding efficiency.

2. MOTION DETECTION AND ANALYSIS

2.1. Overview
One of the main characteristics of any video sequences is the presence of diverse types of motion. Such motion can be produced by static objects with a moving background due to camera motion, objects moving in different trajectories, or simply random motion objects. In these scenarios, the eye gives priority to track fast and unpredictable motion objects (“noticeable motion”) against slow or predictable motion (“non-noticeable motion”). In [7], the authors observe that moving and foreground objects have strong influence on the observer’s attention. For the case of multiple motion types in a scene, noticeable motion attracts more attention. It is assumed that for background or non-noticeable motion objects the viewer perceives just the semantic meaning of the objects holding his/her attention to only noticeable objects. Based on the fact that a viewer gives priority to objects that have noticeable motion, we want to examine what would happen, in terms of coding efficiency and visual quality, if we developed a video coder using these assumptions of human motion attention. We will base our coder on models similar to the approach used in texture based video coding. To do this we use foreground-background distinction. Thus, the motion-classification block in Figure 1 is a foreground-background extraction algorithm.

2.2. Foreground-Background Analysis
The goal of foreground-background analysis is to estimate the camera movement (global motion) that affects both the moving and stationary points of the video sequence. Extracting the background from a video sequence is still an open research problem since various issues such as illumination changes, background object displacements or non-static backgrounds can affect the results. The method also needs to be computationally efficient yet robust. Various issues may degrade the algorithm performance including moving object shadows that appear on background objects, non-static background such as computer screen flicker and so on. In [8, 9, 10, 11] various methods consisting of background substraction by the use of a statistical model describing the background state of each pixel were described. In [12] an algorithm dealing with the detection of moving objects from static background is presented. In this case, a common approach is to model global motion in a scene using a parametric 2-D model. The estimated motion parameters can be used either to compensate the global motion or to extract all macroblocks belonging exclusively to global motion patterns. This estimation technique
is usually a two-step process: selection of the parametric model and optimizing the process. In [13] a perspective model is described that improves the prediction gain over other parametric models. In [14] a pixel-based background segmentation method dealing with fast motion estimation for MPEG-4 video coding is presented.

We implemented for our algorithm the global-local motion compensation scheme presented in [15] by using the following 8-parameter model (an extension of the affine model or perspective model) that provides for panning, zooming, and 3-D rotational motion:

\[
X'_k = \begin{pmatrix} a_0 & x_k + a_1 y_k + a_2 \\ a_1 & y_k + a_0 y_k + a_2 \end{pmatrix}
Y'_k = \begin{pmatrix} a_3 & x_k + a_4 y_k + a_5 \\ a_4 & y_k + a_3 y_k + a_5 \end{pmatrix}
\]

where \((x_k, y_k)\) describes the pixel position of the \(k_{th}\) point before the camera movement (in our case in the reference frame) and \((X'_k, Y'_k)\) contains the corresponding position that has been predicted using the motion vectors, \((mv_x, mv_y)\), (see Equation 2), after the camera movement (in our case in the current frame).

\[
X'_k = x_k + mv_x \\
Y'_k = y_k + mv_y
\]

To determine the eight parameters \(a_0, \ldots, a_7\), in Equation 1, we take \((x_k, y_k)\) as the central points of all background macroblocks (initially all blocks in the current frame are considered as background blocks) and \((X'_k, Y'_k)\) as the corresponding points in the current frame, Equation 2. Hence a CIF \((352 \times 288)\) sequence has \(22 \times 18\) pairs of points, \((x_k, y_k)\) and \((X'_k, Y'_k)\). This leads to an over-complete system that we solved for the eight unknowns using the method of Least Squares (LS).

Iterative Foreground Extraction Algorithm:

1. Once the eight parameters \(a_0, \ldots, a_7\) are obtained, we use the global motion model, Equation 1, on all central points \((x_k, y_k)\) belonging to background blocks of the reference frame. Thus, we obtain \((X''_k, Y''_k)\) representing the ideal positions of the central points in the current frame. Using both sets of points we obtain the difference between \((X''_k, Y''_k)\) and \((X'_k, Y'_k)\) and form \(\delta_k\).

\[
\delta_k = ((X''_k - X'_k)^2 + (Y''_k - Y'_k)^2)^{\frac{1}{2}}
\]

2. \(\delta_k\) is compared with a sequence dependent threshold \(D_{th}\). This is empirically determined for every sequence. Macroblocks with \(\delta_k\) greater than \(D_{th}\) are considered to be foreground, otherwise they are background.

3. Once the new set of background macroblocks are obtained, we remove the isolated macroblocks with a simple morphological closure operation.

4. Finally, after the closure operation we obtain an updated version of the elements belonging to \((x_k, y_k)\) and consequently to \((X'_k, Y'_k)\) that we use to compute the 8-parameter model again using LS.

The algorithm repeats the process until the foreground background configuration no longer changes.

3. TEMPORAL CONSISTENCY, SYNTHESIS, AND INTEGRATION INTO H.264/AVC

The motion models described in the previous section operates on each frame independently of the other frames of the video sequence. This yields an inconsistency in the segmentation across the sequence and can be very noticeable when the video is viewed. To ensure temporal consistency for each group-of-frames, the global motion model is used to warp the background from frame-to-frame as shown in Figure 2. Thus, the side information generator sends to the synthesizer a set of parameters including the global motion parameters, a control flag to indicate which reference frame has been used and the segmentation mask containing the spatial distribution of the background macroblocks. In the synthesizer, it is assumed that the perspective motion model describes the frame-to-frame background displacement. Thus, using the mask segmentation and the motion model it replaces the background in the corresponding position of the current frame by warping towards the ones located in the reference frame indicated by the control flag. The strategy of integrating the proposed method into a conventional video codec is the same as the one used in the texture based video coding [5, 6]. The motion model classification described above was integrated into the H.264/AVC JM 11.0 reference software. In our implementation, 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. I and P frames were conventionally coded; only B
frames were candidates for synthesis since we only performed foreground/background segmentation on the middle frames. The corresponding segmentation mask, motion parameters as well as the control flags were transmitted as side information to the decoder. All macroblocks belonging to a synthesizable background region are labeled as skipped macroblocks. That is, all parameters and variables needed for decoding the macroblocks inside the slice (in decoding order) are set as specified for skipped macroblocks, including the reconstructed YUV samples that were needed for intra prediction, the motion vectors and the reference indices that were used for motion vector prediction. After all macroblocks of a frame are completely decoded, the synthesizer block reconstructs macroblocks belonging to synthesizable background regions and are replaced with the background identified in the corresponding reference frame.

4. EXPERIMENTAL RESULTS

We are interested in determining the performance of the models we tested in terms of data rate savings. To estimate the new data rate for each test sequence, we subtract 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 represent the side information, which is no more than 1.25Kb per frame [5, 6], is then added to obtain the new data rate.

The side information contains the foreground-background 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 (either the first or the last frame of the GOF). In our experiments, we used the following parameters for the Main Profile H.264/AVC codec: Quantization Parameter for intra and inter frames = 16, 20, 24, 32 and 44; 1 reference frame; 3 B frames; CABAC; rate distortion optimization; no interlace; constant channel; 30 frames per second. For the motion classification algorithm we used a Full Search algorithm and a Mean Absolute Error distortion measure to compute the motion vectors. We encoded several sequences to examine the performance of our motion-based approach in various situations and scenarios. We used sequences with several background motion scenarios such as texture backgrounds, zooming, lateral displacements, non-texture backgrounds, and stationary backgrounds. The following video sequences were used based on these criteria: Tabletennis, Waterfall, Mobile, Coastguard, Football. The main reason why we used Tabletennis, Waterfall and Coastguard is that they contain large texture areas in the background so that it was easier to compare and contrast the motion-based approach with the texture-based approach. These sequences also contain background motion effects such as zooming and/or lateral displacements. The Mobile sequence, which is completely inappropriate for the texture based coding since it has no texture areas in the background, allowed us to test the performance of our motion based approach in a scenario where the texture based approach failed.

Our observations from the experiments:

- In general, compared to the texture based video coding method [16, 4, 5, 6], our approach shows an improvement in terms of data rate savings. Such improvement can be seen in various types of sequences such as the (mobile) sequence in which there is no texture parts present in the background; but also in those sequences with large texture parts in the background. Table 1 shows a comparison between our approach and one of the spatial techniques (Gray Level Co-occurrence Matrix + Split and Merge) for the texture-based approach reported in [6]. The data rate savings can be more than 30% depending on the sequence.
- The distinction between foreground and background allows us to label more “candidate” skipped areas (white parts) than the texture based approach, because the texture based approach only identifies texturized objects whereas the motion based method identifies all kinds of background objects, e.g., Figure 3. It can also be seen that the closure operation in the motion classification algorithm removes isolated macroblocks.
- The results in terms of visual quality are very satisfactory, although some “stepping zoom effect” is present in the Waterfall. This is due to the fact that there is a high contrast between the background (the forest) motion and the fast motion of the waterfall, providing an interesting visual artifact when small quantization parameters (16, 20, 24) are used. However, we can assume in general that one focuses on the fast motion of the waterfall instead of the background, and it is hard to perceive the zooming effect unless one carefully looks...
Table 1. Data Rate Savings Obtained for Different Sequences (Compared with the Texture Based Approach). Using a Quantization Parameter of 24.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Original data rate [kb/s]</th>
<th>Data Rate Savings Motion-based</th>
<th>Data Rate Savings Texture-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tabletennis</td>
<td>840.49</td>
<td>17.56%</td>
<td>15.22%</td>
</tr>
<tr>
<td>Waterfall</td>
<td>2399.31</td>
<td>35.79%</td>
<td>16.02%</td>
</tr>
<tr>
<td>Mobile</td>
<td>2158.66</td>
<td>15.42%</td>
<td>16.02%</td>
</tr>
<tr>
<td>Coastguard</td>
<td>2593.82</td>
<td>27.19%</td>
<td>16.28%</td>
</tr>
<tr>
<td>Football</td>
<td>1962.14</td>
<td>18.16%</td>
<td>12.23%</td>
</tr>
</tbody>
</table>

for it (Figure 4). Similarly, when local motion is present in the background such as the water in the Coastguard sequence the viewer focuses on the foreground ships rather than in the water movement making the absence of “natural” local motion in the water of the reconstructed sequence not easy to perceive. Note that, when the synthesizer does not keep the reconstruction process below visual perception limits, visual artifacts may appear either in the texture based or in the motion based system.

5. CONCLUSIONS

The contribution of this paper is the description of an approach for video coding based on human visual motion perception instead of texture properties. Methods to separate the foreground from the background were investigated and were incorporated into a conventional video codec (e.g. H.264) where the regions modeled by the global motion model were not coded in an usual manner. We have shown that we can reduce the data rate by more than 15% when compared with more classical approaches such as H.264.

6. REFERENCES


