Global/Local Motion-Compensated Frame Interpolation for Low Bitrate Video

Shan Liu, Zhidong Yan, JongWon Kim and C. -C. Jay Kuo
Integrated Media Systems Center
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2564 Email: {shanl, zhidong, jongwon, cckuo}@sipi.usc.edu

ABSTRACT

A new motion-compensated frame interpolation scheme for low bitrate video based on the ITU-T H.263/H.263+ standard is investigated in this research. The proposed scheme works solely on the decoded bitstream with a block-based approach to achieve interpolation results. It is composed of two main modules: the background/foreground segmentation module and the hybrid motion compensated frame interpolation module. The background/foreground segmentation module uses a global motion model to estimate the background motion and an iterative background update to refine the segmentation. The hybrid motion compensated frame interpolation module is employed to reconstruct background and foreground, respectively. Global motion compensation and frame interpolation is applied to background blocks where either the 6-parameter affine or the 8-parameter perspective model is used to reduce the computational complexity and implement perspective correction, while local motion compensation and frame interpolation with localized triangular patch mapping is applied to the foreground area. Experiments show that the proposed scheme can achieve higher overall visual quality compared to conventional block-based frame interpolation schemes.

Keywords: hybrid motion compensation, frame interpolation, video post-processing, and affine/perspective global motion compensation.

1 Introduction

Due to the limited bandwidth and storage space, a low bitrate video encoder may not encode all frames in an image sequence. A frame skipping technique, i.e. temporal sub-sampling, is commonly used to achieve the target compression ratio in very low bitrate video conferencing applications. As a result, temporal domain artifacts such as motion jerkiness significantly degrade the visual quality of decoded video. The frame interpolation technique is introduced at the decoder as a post-processing tool to reconstruct skipped frames and reduce temporal artifacts.

A number of frame interpolation schemes have been proposed recently. Simple frame repetition [1] suffers jerky object motions. With frame averaging [1], jerky motions are smoothed at the cost of ghosts. Linear interpolation [2] generates blurred moving areas because pixels of different objects are improperly mixed. To avoid these problems, a detailed description of motion-compensated frame interpolation (MCI) was introduced by Musmann [3], where motion estimation and object segmentation are separated. Chen *et al.* [4] proposed an advanced scheme by combining the background/foreground segmentation and MCI together. However, it mainly focused on the talking head sequence and a specific mesh model would be needed for each video, which increases the coding overhead and the computational complexity for mesh mapping. Thomas [5] proposed a MCI scheme that focused on both covered/uncovered backgrounds and applied a hierarchical motion estimation for interpolation. This technique can be generally combined with other frame interpolation techniques to achieve a better result.

Two types of motion fields, pixel-based and block-based, are typically used to provide the motion trajectory between current and previous frames. The pixel-based motion field, while results in more accurate interpolation, demands a high computational complexity. In contrast, the block-based field requires only one motion vector for each block, which has been adopted by most video compression standards such as ITU-T H.263/H.263+ (where the block size is 16×16). It provides acceptable visual quality with a carefully designed algorithm. By employing block-based motion field, Kuo *et al.* [7] proposed an MCI scheme, called the deformable block-based fast MCI (DB-FMCI), which demonstrates a good performance-complexity tradeoff. However, it exhibits blocking artifacts in the motion background area, since the motion background is treated in the same fashion as the foreground. That is, only local motion compensation (LMC) is applied to both the foreground and the background, which tends to result in a global motion interpolation error. By following the framework of DB-FMCI [7], we investigate an enhanced approach by employing global/local motion compensation in this research. However, since the algorithm is used as an encoder-independent post-processing unit, we are constrained by the decoded block motion vectors provided by the encoder. Thus, we face the uncertainty of compression-optimized block motion vectors, and have to settle at a compromised solution for the selected type of video.

The proposed MCI technique is developed in two main modules: the background/foreground segmentation module and the hybrid motion compensated frame interpolation module. First, the block-based motion segmentation technique is developed to iteratively divide a frame into background and foreground blocks so that it can be tied with global/local motion estimation/compensation. Especially, to cope with the uncertainty of compression-optimized block motion vectors, it is combined with the intensity-based HVS (human visual system) segmentation of FMCI [7]. Next, the hybrid motion compensated frame interpolation module is employed to reconstruct the background and the foreground, respectively. Global motion compensation and frame interpolation are applied to background blocks where either the 6-parameter affine or the 8-parameter perspective model is used to reduce the computational complexity and achieve perspective correction. Local motion compensation and frame interpolation with localized triangular patch mapping are used for the foreground area. By assuming a constant motion velocity between neighboring frames, the proposed FMCI is capable of inserting as many frames as needed between decoded frames without requiring extra bits. In addition, the proposed MCI scheme does not require any bitstream syntax change of ITU-T H.263/H.263+.

It is worthwhile to point out that research on global/local motion estimation/compensation has been traditionally considered at the encoder end to achieve a higher compression ratio, e.g. [9], [10], [11]. Had global estimation done at the encoder with the notion of post-processing, it should provide a further performance gain so that the resulting MCI may overcome the sequence type barrier.

The paper is organized as follows. The overview of the proposed system is described in Section 2. The background/foreground segmentation technique is detailed in Section 3. The hybrid motion compensation/interpolation methods for the background and the foreground are discussed in Section 4. Experimental results are presented in Section 5. Concluding remarks and future work are given in Section 6.

2 Overview of the Proposed System

The proposed hybrid motion-compensated frame interpolation system with perspective correction on background blocks is shown in Fig. 1. It consists of two main modules, i.e. the background/foreground segmentation module and the hybrid motion compensated frame interpolation module. The proposed scheme works solely on the decoded bitstream by using a block-based approach. It achieves interpolation results independently of the encoder except for the use of block motion vectors provided by the encoder. The use of available motion vectors is a key design factor, since it fundamentally limits the quality of interpolation on one hand while relieving the decoder from the burden of extensive and time-consuming motion search on the other hand. The proposed system will finetune inconsistent block motion vectors, and use them to achieve an region-adaptive motion compensated interpolation.

In the proposed system, frames and associated motion vectors of a given sequence are first obtained from the H.263 decoder. The sequence contains independent I frames and dependent P frames along with motion vectors for each P frame. First, the background/foreground segmentation algorithm is applied to a pair of frames to be

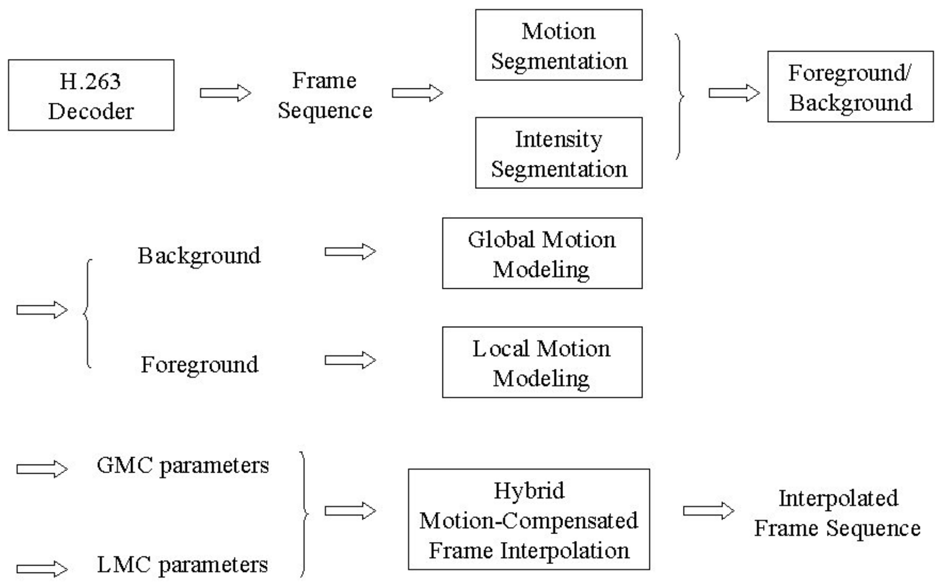

Figure 1: The proposed motion-compensated frame interpolation system consisting of two main modules.

interpolated. As a result, blocks in each frame can be divided into two types: background and foreground blocks. A hybrid motion compensation algorithm is then applied to the two different types of blocks. Background blocks are processed using global motion compensation (GMC), where perspective correction can also be achieved by using the 8-parameter motion model. Foreground blocks are treated with traditional local motion compensation. GMC or LMC parameters for each block are obtained, and skipped frames are reconstructed (i.e. interpolated) in the hybrid motion-compensated frame interpolation module. The detailed design and implementation of these two modules will be described in the following sections.

3 Background/Foreground Segmentation

Without segmentation, each frame is viewed as one whole unit so that the motion object is treated the same as the background. As a result, the visual quality of the interpolated frame is usually not satisfactory in a block-based frame interpolation scheme. The objective of background/foreground segmentation is to divide the frame into background and foreground blocks and treat them differently.

The segmentation method proposed by Kuo *et al.* [7] exploits the human visual system (HVS), where pixels are determined to be either foreground or background by their intensity change between successive frames. It works well for sequences with stationary background. However, since it treats the motion background the same as the foreground object, it compensates the background motion region locally, which tends to introduce noticeable blocking artifacts in the background area. Here, we propose a motion based segmentation method that can efficiently separate the object from the motion background. Combined with the complementary intensity-based HVS segmentation, the proposed segmentation scheme greatly achieve better visual quality due to the reduction of background artifacts.

3.1 Motion Models

Generally speaking, motion arising in an image sequence is a combination of global and local motion activities. Global motion is usually introduced by camera's operation and/or movement, while local motion comes from the displacement of objects in the scene. To separate the foreground from the background, it is important to estimate and compensate camera's motion. Algorithms for this separation differ in the model used to represent motion as well as the model parameter estimation technique. Usually, two sets of points, (x_i, y_i) and (x'_i, y'_i), $i = 1, \ldots, N$, are chosen to calculate the model parameters, where $(x_i,\, y_i)$ specifies the pixel position of the i_{th} selected point before camera movement and $(x'_i,\, y'_i)$ is the corresponding position after camera movement. Among the 2-, 4-, 6- and 8-parameter models, 2- and 4-parameter models usually lack the desired quality since the 2-parameter model only estimates the panning while the 4-parameter model provides additional 2-D scaling. The following 6- or 8-parameter model is our choice to represent the background motion.

The 6-parameter model (or called the affine model) is able to capture the panning, zooming and 2-D rotational motion. They are expressed as

$$x'_i = a_0 \times x_i + a_1 \times y_i + a_2, y'_i = a_3 \times x_i + a_4 \times y_i + a_5.$$

To solve the six unknowns $a_0, \cdots, a_5$ in the above system, three independent points would be sufficient. However, the coordinates may contain measurement errors. Therefore, it is common to add more points to improve the robustness and solve the resulting over-determinant system for the least squares (LS) solution. The system of equations with N points can be easily represented by

$$Ka = U,$$

where

$$K = \begin{pmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \cdots & \cdots & \cdots \\ x_N & y_N & 1 \end{pmatrix}, \quad a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}, \quad U = \begin{pmatrix} x'_0 \\ x'_1 \\ x'_2 \\ \cdots \\ x'_N \end{pmatrix}.$$

The LS solution can be written as

$$a = \left(K^t K\right)^{-1} \left(K^t U\right).$$

In a similar way, we can obtain a_3, a_4 and a_5. Since $K^t K$ is a 3×3 matrix which is generally non-singular, the computational complexity is low. In our approach, the affine model is adopted for both global and local motion compensation applications. Even though it often fails to capture depth and local deformations, it serves well as a compact and fast alternative for the whole background. For foreground blocks, DB-FMCI relies on the affine model for its flexible and powerful localized motion compensation.

We adopt the 8-parameter model (i.e. the perspective model) to estimate panning, zooming, and 3-D rotational motion. It can be expressed as

$$x'_i = \frac{a_0 \times x_i + a_1 \times y_i + a_2}{a_6 \times x_i + a_7 \times y_i + 1}, y'_i = \frac{a_3 \times x_i + a_4 \times y_i + a_5}{a_6 \times x_i + a_7 \times y_i + 1}$$

Several MPEG4 test sequences such as "Foreman" and "Container Ship" do contains the global motion that can best be represented by panning, zooming and additional 3-D rotation. Thus, in this paper, the 8-parameter model is alternatively chosen to compensate the global motion. Four points would be sufficient to solve the above system of equations. However, similar to the 6-parameter model, it is desirable to adopt the following LS solution for enhanced robustness:

$$Ka = U,$$

where

$$K = \begin{pmatrix} X_0 & y_0 & 1 & 0 & 0 & 0 & -x'_0 \times X_0 & -x'_0 \times y_0 \\ 0 & 0 & 0 & X_0 & y_0 & 1 & -y'_0 \times X_0 & -y'_0 \times y_0 \\ X_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 \times X_1 & -x'_1 \times y_1 \\ 0 & 0 & 0 & X_1 & y_1 & 1 & -y'_1 \times X_1 & -y'_1 \times y_1 \\ X_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2 \times X_2 & -x'_2 \times y_2 \\ 0 & 0 & 0 & X_2 & y_2 & 1 & -y'_2 \times X_2 & -y'_2 \times y_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ X_N & y_N & 1 & 0 & 0 & 0 & -x'_N \times X_N & -x'_N \times y_N \\ 0 & 0 & 0 & X_N & y_N & 1 & -y'_N \times X_N & -y'_N \times y_N \end{pmatrix} \quad a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \quad U = \begin{pmatrix} x'_0 \\ y'_0 \\ x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \cdots \\ x'_N \\ y'_N \end{pmatrix}$$

The LS solution can be found via

$$a = \left(K^t K\right)^{-1} \left(K^t U\right).$$

It is worthwhile to point out that, in most experiments, the 8×8 matrix $\mathbf{K^t K}$ is singular so that the normal matrix inverse function does not work properly. Thus, the Singular Value Decomposition (SVD) method [12], which can be viewed as a generalized matrix inverse algorithm, was adopted to solve this problem. Compare to the 6-parameter model, the 8-parameter model provides perspective correction for background areas and achieves better visual quality, while introduces a higher computational complexity at the same time.

3.2 Iterative Motion Segmentation

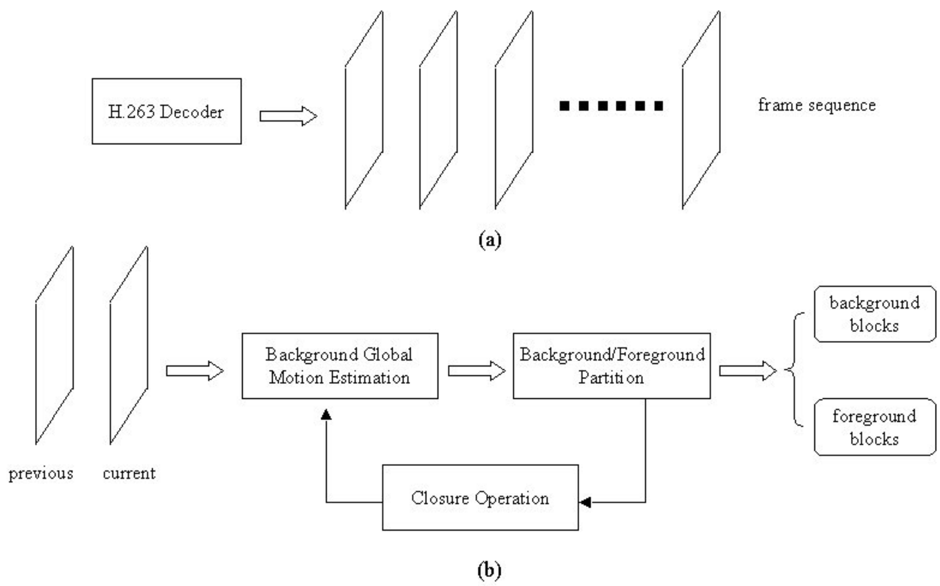

Figure 2: (a) The frame sequence generated by H.263 decoder, and (b) the iterative background/foreground segmentation scheme.

Fig. 2(a) shows the frame sequence generated by H.263 decoder, which is usually in the IPPP... format. The objective of frame interpolation is to insert one or more frames between two adjacent frames. Fig. 2(b) shows iterative background/foreground segmentation for partitioning the background and foreground blocks in the previous frame and the current frame. In the H.263 decoded frame, each MB (macroblock, in 16×16 size) is represented by a motion vector associated with the central point. By exploiting the global motion model obtained from motion correspondence between the current and previous frames, frames can be iteratively partitioned into background/foreground blocks as follows.

- Step 1: Initially, all blocks in the current frame are considered as background blocks.
- Step 2: The central points of all background blocks are taken as $(x_i,\, y_i)$ in the global motion model. The corresponding points in the previous frame, which are obtained from $(x_i,\, y_i)$ and the decoded motion vectors, are taken as $(x'_i,\, y'_i)$. The 6- or 8-parameters can be calculated accordingly. For a normal QCIF sequence, the frame has a size of 176×144 pixels. The resulting 16×16 blocks provides 11×9 pairs of points $(x_i,\, y_i)$ and $(x'_i,\, y'_i)$, which are enough for the LS calculation.
- Step 3: The calculated global motion parameters $(a_0, a_1, \cdots a_8)$ describes an estimation of the camera motion. By applying the global motion model on all central points $(x_i,\, y_i)$ in the background blocks of the current frame, we can obtain a set of points $(X"_i,\, Y"_i)$. $(X"_i,\, Y"_i)$ represents ideal positions of these central points in the previous frame under the estimated camera motion.

- Step 4: Since the global motion model is only an estimation of the camera motion, it is not suitable to represent the motion activities of foreground blocks and also has a minor difference from the true background motion. The difference between $(X"_i, Y"_i)$ and (x'_i, y'_i) are calculated in the form of the distance between them

$$d_i = \sqrt{(X"_i - x'_i)^2 + (Y"_i - y'_i)^2}$$

Distance d_i is compared with a predefined threshold D. If d_i is greater than D, it is likely that the corresponding block is not a background block so that we remove it from the set of background blocks. On the contrary, if d_i is less than D, the corresponding block is still considered as a background block. After this process for all background blocks, a new set of background blocks can be obtained.

- Step 5: To reduce block artifacts, we attempt to remove isolated background and foreground blocks. This is achieved by adopting the closure operation on all blocks. Basically, each block has eight direct neighboring blocks. The closure operation exploits the neighboring block information and decides whether to make a change on the current block. The 8 neighboring blocks of block B_0 are represented by $B_1, B_2, \cdots, B_8$. We denote the number of background and foreground blocks in these 8 blocks by N_b and N_f, respectively. When B_0 is a foreground block, if most of its neighbors are background blocks, i.e. $N_f < N_b$, then B_0 will be changed to a background block. When B_0 is a background block, if most of its neighbors are foreground blocks, i.e. $N_f > N_b$, then B_0 will be changed to a foreground block. In all other cases, B_0 remains unchanged.

  The closure operation is performed on the whole frame in the raster-scan order (i.e. from left to right and top to bottom). It is repeated until either the operation converges (i.e. no more changes on the blocks) or it reaches the maximum number of iterations. In the experiment, the maximum number of iterations is set to 3.

- Step 6: After the closure operation, an updated set of background blocks can be obtained. The central points of all background blocks are taken as (x_i, y_i) in the global motion model. Similarly, the corresponding points in the current frame are taken as (x'_i, y'_i). The global motion parameters can then be calculated again. We repeat Steps 2-5 until the background/foreground distribution is no longer changed. As a result, we obtain the final segmentation result of background and foreground blocks.

The segmented background and foreground blocks in an image frame as well as their motion vectors are illustrated in Fig. 3. Compared with most of other video segmentation techniques, this technique is rough. It does not give very accurate object segmentation due to its block-based nature. Hence, a combination of block based motion segmentation and pixel level intensity segmentation is adopted here to reduce the blocking artifacts and improve the segmentation result.

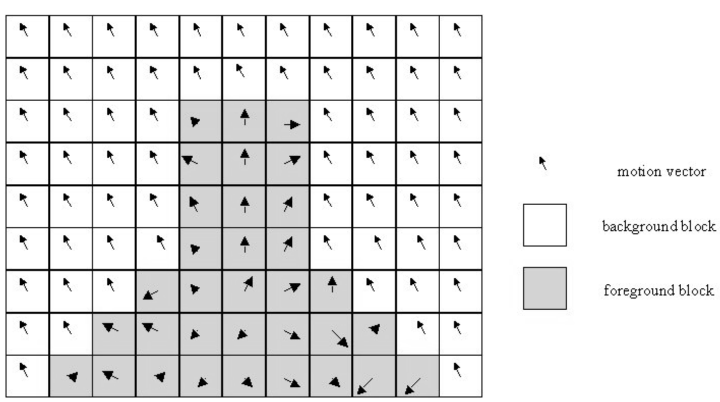

Figure 3: Illustration of the motion segmented background/foreground blocks inside a frame.

3.3 Complementary Intensity Segmentation

Noticeable blocking artifacts can be observed if only block-based motion segmentation is used to identify foreground and background blocks. On the other hand, intensity-based segmentation alone cannot detect motion background from the foreground object. Thus, we propose a segmentation scheme which integrates these approaches to provide a simply yet accurate segmentation results which is sensitive to the background motion.

The intensity based segmentation method follows the framework proposed by Kuo *et al.* [7]. It consists of three major steps: moving object segmentation, morphological closure operation and HVS segmentation. In moving object segmentation, pixel intensities are compared between the previous frame and the current frame. If the difference is greater than a pre-defined small threshold, the pixel is considered as an object pixel. Otherwise, it belongs to the stationary background. Then, the morphological closure operation is used to remove small holes inside the segmented object. Finally, each 4×4 square is matched with some pre-defined blocks based on HVS and is replaced with the most similar HVS block. Therefore, the segmentation accuracy can reach the 4×4 pixel level.

Once motion segmentation and intensity segmentation steps are done separately, the logical *and* operator is applied to these two results. If a pixel belongs to a moving object in HVS segmentation and a foreground block in motion segmentation at the same time, it is said to be a foreground pixel. If it belongs to a moving object in HVS segmentation and a background block in motion segmentation, it is considered as a part of the motion background. If it belongs to background in both HVS and motion segmentations, it is said to be stationary. Pixels belonging to the moving object by HVS segmentation but in background areas by motion segmentation are possibly due to boundary errors caused by the blocking effect. It is demonstrated by experimental results that the proposed segmentation scheme can work efficiently and effectively. The segmentation result of two successive frames in the Foreman sequence is shown in Fig. 4.

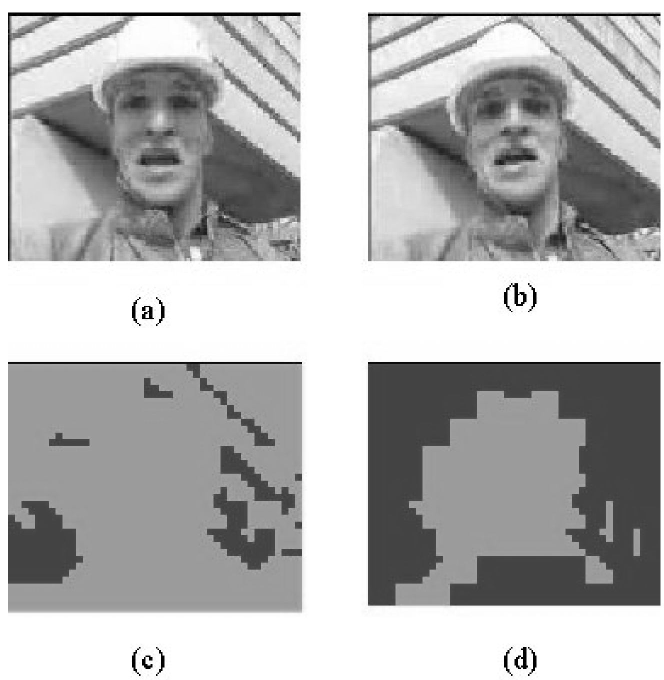

Figure 4: The background/foreground segmentation result of the Foreman sequence: (a) the previous frame (66_{th}), (b) the current frame (71_{th}), (c) the HVS segmentation result, and (d) the final background/foreground segmentation result.

4 Hybrid Motion-Compensated Frame Interpolation

For hybrid motion-compensated frame interpolation, background motion is estimated by using the global motion model. However, unlike the global motion model employed in the background/foreground segmentation, where the current and the previous frames provide (x_i, y_i) and (x_i', y_i') point sets mainly to determine the background and the foreground blocks, the global motion model adopted here estimates the corresponding global motion parameters from the interpolated frame to the previous and current frames for bidirectional interpolation. These parameters will be used to reconstruct background pixels in the interpolated frames. The motion vectors of foreground blocks are estimated by using the local 6-parameter model, in which a set of 6-parameters will be applied to the foreground pixel construction in an interpolated frame.

- Step 1: Calculate the coordinates of key point sets.

Let us denote the previous frame by F_p, the current frame by F_c, and the frame to be interpolated by F_i. The central points of all blocks in F_c are chosen as the basic point set used in the motion compensation. They are denoted by

$$SV_c = \{V_{c1}, V_{c2}, \cdots, V_{cN}\},$$

where N is the total number of central points. Some points are associated with background blocks while others with foreground blocks. Let SVB_c and SVF_c denote the background/foreground point sets, respectively. Then, we have $SV_c = SVB_c + SVF_c$.

In order to estimate the motion between two frames, at least 4 (or 3) point pairs are needed for the 8- (or the 6-) parameter model. For each point V_{cj}, $j = 1, \cdots$, the corresponding point in the previous frame, denoted by V_{pj}, can be calculated by using decoded motion vectors. All such points in the previous frame form a set $SV_p = \{V_{p1}, V_{p2}, \cdots, V_{pN}\}$. Similarly, SVB_p and SVF_p are used to specify the background and foreground point sets, and we have $SV_p = SVB_p + SVF_p$. It is assumed that the motion velocity from the current frame to the previous frame is constant. This constant motion velocity assumption can provide a reasonable estimation of the corresponding point V_{ij} in the interpolated frame, which is linearly interpolated between V_{pj} and V_{cj}. Consequently, we obtain $SV_i = \{V_{i1}, V_{i2}, \cdots, V_{iN}\}$ and $SV_i = SVB_i + SVF_i$. The constant motion velocity assumption is used only to estimate the vertex point V_{ij}, which is needed in the background and foreground motion estimation and compensation. Since linear interpolation is too simple, this assumption should not be extended to all remaining pixels in the interpolated frame. More accurate pixel interpolation should be implemented based on the 6 and 8-parameter models.

- Step 2: Apply global/local motion models and calculate global/local motion parameters.

With the corresponding coordinates of points between the interpolated frame and previous/current frame, we are able to calculate both global and local motion parameters and build the proper motion models. For motion object, i.e. in the foreground region, the localized 6-parameter affine model is applied and the triangular patch mapping method proposed in [7] is adopted. With this approach, each set of four neighborhood central points which form a square is divided into two triangles by using the shorter diagonal criterion. Three vertices of each triangular provide a unique solution to a localized 6-parameter model, which is used to generate all pixels in a local region in the interpolated frame.

Both 6-parameter and 8-parameter models can be applied to estimate the global motion based on different user demands. Unlike the foreground motion model, which is localized, all background pixels follow the same motion which is represented by one global motion model. The 6- or 8-parameter global motion models can be build by using the least-square estimation solution as discussed before.

- Step 3: Construct interpolated frames with global/local motion compensation.

We use bidirectional interpolation so that each pixel in the interpolated frame is constructed by the combination of backward and forward interpolation. Consider a pixel in the interpolated frame. With the backward approach, its corresponding location in the previous frame is calculated and its value is obtained. With the forward approach, the corresponding pixel location in the current frame is estimated and its value is taken. These two values are linearly weighted according to the distance between the interpolated frame and the previous/current frame and added to determine the corresponding pixel value in the interpolated frame. Depending on whether a pixel belongs to a background or a foreground block, motion parameters used to interpolate its value are different.

Since all background blocks undergo the same motion, only one set of 6- or 8-parameters is needed to represent this background motion. If a pixel belongs to the background, its backward interpolated value can be determined by the backward motion compensated interpolation, where the global motion parameters are calculated by SVB_p and SVB_i. SVB_i and SVB_c can be used to calculate the 6 or 8-parameters to reflect all background blocks' motion between the interpolated frame and the current frame, which are used to calculate the forward interpolated pixel value.

If a pixel belongs to a foreground block, it has its own motion which is different from that of other blocks. Since different foreground blocks have different motions, the localized foreground motion parameters are generally not the same. Therefore, the backward and forward estimated value of the interpolated pixel has to be calculated according to the local motion model associated with the triangular patch it belongs to. For more details, we refer to [7].

5 Experimental Results

For performance evaluation, we have applied the proposed scheme to two test sequences of the QCIF format: Miss America and Foreman. Especially the visual quality is extensively compared that of DB-FMCI to verify the additional gain due to extra global/local hybrid processing. The results are shown in Fig. 5 and Fig. 6, respectively.

In the Miss America QCIF video sequence, the original frame rate of input sequence is 30 frame per second (fps) in the encoder end, the basic mode (i.e. no optional mode is activated) is selected, and the quantization step 20 and frame skip 10 are used. The required bandwidth for the encoded bitstream is only 8kbps due to the adoption of a large frame skip. As a result, from the decoder end, we can obtain a frame rate of 3fps. 9 frames are inserted between two adjacent frames in the decoded bitstream to recover the frame rate back to 30fps. Fig. 5 (a) shows the 61_{th} frame in the original sequence, (e) shows the 71_{th} frame in the original sequence. We compare the interpolated 66_{th} frames from the 61_{th} and the 71_{th} frames by using different frame interpolation methods in the same figure, where (b) is obtained by DB-FMCI and (d) is obtained using the proposed scheme. For comparison, we show the original 66_{th} frame in (c). From them, we see that (b), (c), (d) are almost the same, no visual differences is observable. This is due to the fact that the background motion in the Miss America sequence is too small to be detected. Thus, even though we apply the global motion compensation to the background, it is not noticeable.

In the Foreman QCIF sequence, the original frame rate of the input sequence is also 30 frame per second (fps) in the encoder end, the basic mode (i.e. no optional mode is activated) is selected, and the quantization step 13 and frame skip 4 are used. The required bandwidth for the encoded bitstream is 34.36kbps. From the decoder end, we can obtain a frame rate of 5.93fps. 4 frames are inserted between two adjacent frames in the decoded bitstream to recover the frame rate back to 30fps. Fig. 6 (a) shows the 66_{th} frame in the original sequence, (e) shows the 71_{th} frame in the original sequence. Similarly, we show the interpolated 68_{th} frames from the 66_{th} and the 71_{th} frames by using different frame interpolation methods in the same figure, where (b) is obtained by DB-FMCI, (d) is obtained using the proposed scheme, and (c) is the original 66_{th}. For this case, we see a clear difference in the background area in (b), (c) and (d). With the DB-FMCI approach, the background is blurred, while the blurred regions are removed in the proposed scheme with the affine global motion compensation. (d) has a significant visual quality improvement over (b). However, both (b) and (d) do not achieve as good results as the original frame. Some background motion is still not compensated well and requires further postprocessing. This is still under our current research and development.

By replacing affine GMC with perspective GMC, visual enhancement in background areas can be achieved at

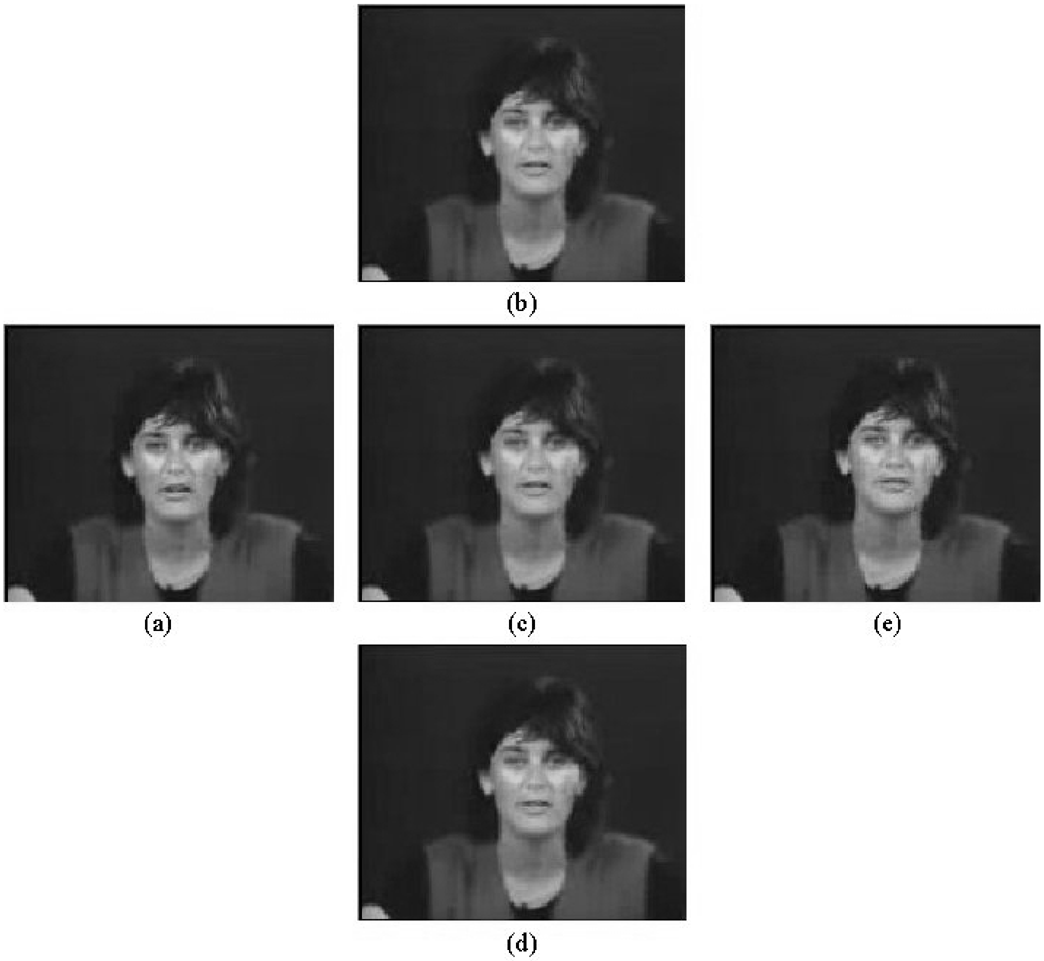

Figure 5: Comparison of different interpolation methods on Miss America sequence, where (a), (c), (e) show the 61_{th}, the 66_{th}, and the 71_{th} frames in the original sequence, respectively, and (b), (d) show the 66_{th} frame obtained by interpolating between the 61_{th} and the 71_{th} frames by using DB-FMCI and the proposed scheme, respectively.

the cost of a higher computational cost. As shown in Fig. 7, we have the interpolated 95_{th} frames from affine GMC and perspective GMC. The background region interpolated by perspective GMC looks better than that of affine GMC and inconsistency occurring in the upper right face area in the affine GMC frame is fixed by the perspective GMC method.

6 Conclusion and Future Work

In this research, we proposed a new scheme for the motion-compensated frame interpolation for low bitrate video with perspective correction. The proposed scheme mainly contains two main modules, i.e. the background/foreground segmentation module and the hybrid motion compensated frame interpolation module. The foreground can be segmented from both still and motion background efficiently in the segmentation module. They are then treated with different motion models to achieve the motion-compensated frame interpolation. The background motion is represented by a 6- or 8-parameter global motion model to achieve perspective correction at a low computational complexity, while foreground motions are modeled by using localized affine 6-parameters. One set of global motion parameters is applied to all background blocks for interpolation, while each foreground block uses its own motion parameters to construct the interpolated pixels in the corresponding triangular patch. Experi-

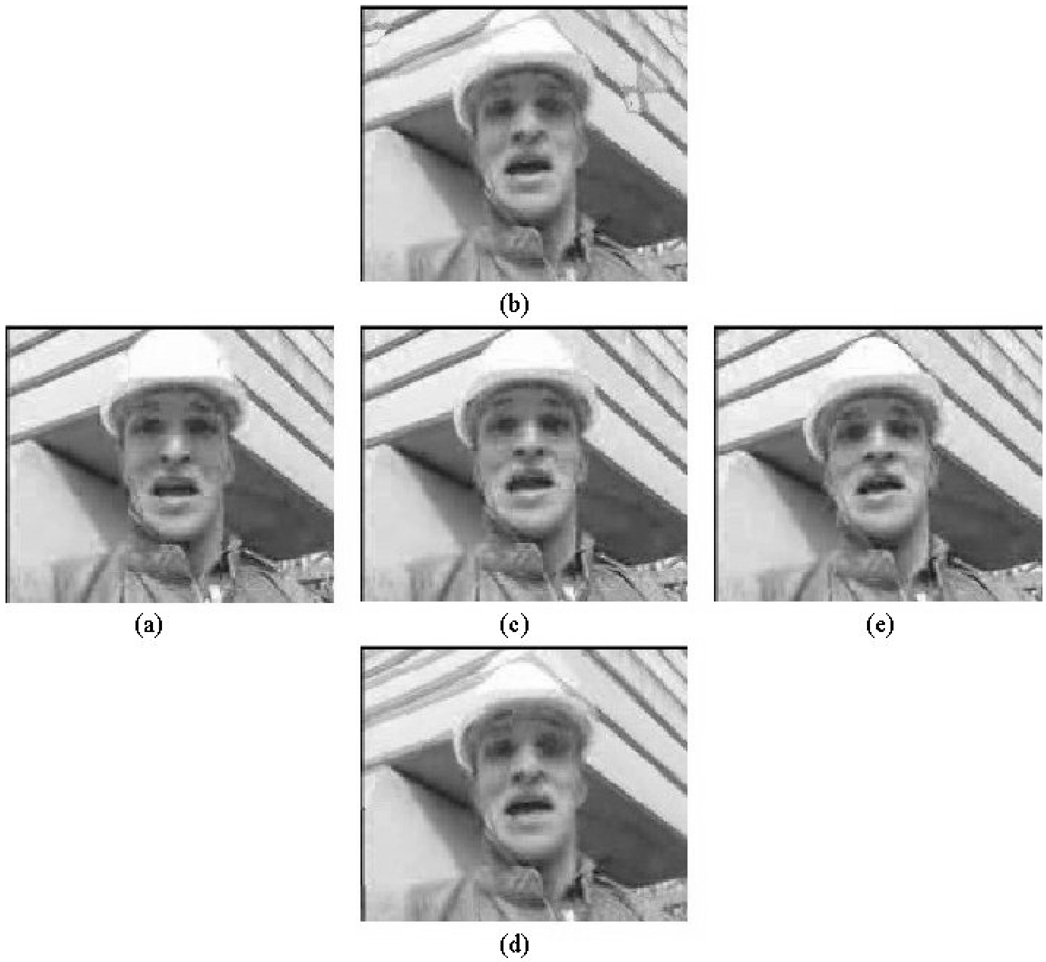

Figure 6: Comparison of different interpolation methods applied to the Foreman sequence, where (a), (c), (e) show the 66_{th}, the 68_{th}, and the 71_{th} frames in the original sequence, respectively, (b) and (d) show the 68_{th} frame obtained by interpolating between the 66_{th} and the 71_{th} frames by using DB-FMCI and the proposed scheme, respectively.

ments show that the proposed scheme works better in the motion background area compared that of the DB-FMCI method [7]. A significant visual quality enhancement can be achieved to avoid blocking artifacts. For the small or zero motion background, there is no noticeable visual difference between the two schemes.

In the motion compensation module, we adopt a constant motion velocity constraint to estimate the key point locations in the interpolated frame. They are used in both the 8- and the 6-parameter models. However, it may not be always true that the motion is in constant velocity. Errors generated in key points cannot be recovered in the later model estimation procedure. We are working on an improved method, which will consider the nonlinear interpolation from the current frame to the previous frame rather than the linear interpolation of motion vectors. The performance on the foreground area should also be improved by introducing 8-parameter model to foreground motion compensation at a higher computational cost.

7 REFERENCES

[1] A. M. Tekalp, *Digital video processing*, Prentice hall, NJ, 1995.

(a) (b)

Figure 7: Comparison of different interpolation methods applied to the Foreman sequence. (a) the interpolated 95_{th} frame by affine GMC and (b) the interpolated 95_{th} frame by perspective GMC.

[2] A. N. Netravali and J. D. Robbins, "Motion-adaptive interpolation of television frames", in *Proc. Picture Coding Symposium*, 12.1, June, 1981.

[3] H. G. Musmann, P. Pirsch and H. J. Grallert, "Advances in picture coding", *Proc. IEEE*, vol. 73, pp. 523-548, Apr. 1985.

[4] T. Chen, Y. Wang, H. P. Graf, and C. Swain, "A new frame interpolation scheme for talking head sequences", in *Proc. ICIP-95*, vol. 2, pp. 591-594, 1995.

[5] R. Thomas and M. Bierling, "Motion compensating interpolation considering covered and uncovered background", *Signal Processing: Image Communication*, pp. 191-212, 1989.

[6] C. K. Wong and O. C. Au, "Fast motion compensated temporal interpolation for video", in *Proc. SPIE Visual Communications and Image Processing '95*, pp. 1108-1118, May 1995.

[7] T. Kuo and C. -C. Jay Kuo, "Motion-compensated interpolation for low-bit-rate video quality enhancement", in *Proc. SPIE Applications of Digital Image Processing*, vol.3460, pp.277-288, 1998.

[8] S. F. Wu and J. Kittler, "A differential method for simultaneous estimation of rotation, change of scale and translation", *Signal Processing: Image Communication*, vol. 2, no. 1, pp. 69-80, May 1990.

[9] F. Moscheni, F. Dufaux and M. Kunt, "A new two-stage global/local motion estimation based on a background/foreground segmentation", *Proc. ICASSP-95*, vol. 4, pp. 2261-2264, 1995.

[10] Hirohisa Jozawa, Kazuto Kamikura, Atsushi Sagata, Hiroshi Kotera and Hiroshi Watanabe, "Two-Stage Motion Compensation Using Adaptive Global MC and Local Affine MC", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 75-85, Feb. 1997.

[11] Chung-Tao Chu, Dimitris Anastassiou and Shih-Fu Chang, "Hierarchical Global Motion Estimation/Compensation in Low Bitrate Video Coding", in *Proc. IEEE International Symposium on Circuits and Systems*, June 1997.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical recipes in C: the art of scientific computing*, pp. 60-71, Cambridge University Press, 1993.