FRAME INTERPOLATION USING TRANSMITTED BLOCK-BASED MOTION VECTORS

Seung-Chul Yoon and Narendra Ahuja

Department of Electrical and Computer Engineering
Beckman Institute for Advanced Science and Technology
University of Illinois, Urbana, IL 61801
scyoon@vision.ai.uiuc.edu

ABSTRACT

The proposed technique is designed for interpolating video frames often dropped during compression using standards, such as MPEG-4 and H.263 at very low bit rates. The coarse motion vector field is refined at the receiving side using mesh-based motion estimation instead of using computationally intensive dense motion estimation. We propose a framework for detecting and utilizing local motion boundaries in terms of an explicit model. Motion boundaries are modeled using edge detection and Hough transform. The motion of the occluding side is represented by affine mapping. The newly appearing region is also detected. Each pixel is interpolated differently according to adaptive interpolation based on the property of the pixel: moving, static, and disoccluded.

1. INTRODUCTION

Block-based motion estimation produces blockwise constant motion vectors. The analysis of the coarse block-based motion vector field provides very limited information about motion boundaries. In video compression standards, a decompressed motion vector field provides little information about motion boundaries because it says only which blocks might have motion boundaries at their block boundaries. In reality, the block boundaries do not correspond to the actual motion boundaries. Previous frame interpolation approaches have treated the block boundaries as *noise* and have not fully exploited the property of the blocky motion vector field.

In this paper we propose a new technique for frame interpolation of video sequences. We use a model of motion discontinuities. The model exploits a structure of the already given motion vector field. The model also approximates the motion discontinuities. With this model, we can predict the newly appearing (i.e., the disoccluded) pixels so that these pixels can be used to generate a background sprite during interpolation. This model also provides us with a solution for the foreground/background segmentation problem.

Block-based motion vectors have been used for frame interpolation but their results have been disappointing due to the blockwise coding of motion vectors and residuals. To solve this problem we use a mesh-based method for estimating motion in the decoder. Adopting a mesh topology, we define the neighborhood of each node (interchangeably, vertex) given the block-based motion vectors. Affine mapping of a triangular mesh provides a motion vector for each pixel of the triangle. We use a hierarchical method for estimating nodal displacements. We exploit the neighborhood of blockwise constant motion vectors to refine the nodal displacement. For this goal, we use a pel-recursive motion estimation algorithm. Biemond *et al.* [1] formulated a recursive Wiener estimate for pel-recursion. The recursive equation is obtained by considering a Taylor series expansion of the intensity image as a function of the motion vector to be estimated. It is well known that the inherent linearization due to Taylor series representation limits the applicability of the algorithm to situations with small motion and where a good initial estimate of motion is available. From this observation, we can use a block-based motion vector field for a good initial guess. Looking at motion compensation errors and a topology of triangulation, we detect motion boundaries, moving foreground, and background. The motion boundaries are modeled using strong edges and a Hough transform.

Previous approaches for frame interpolation need to perform separate motion estimation, such as pel-recursion [2] and mesh-based warping [3]. Others have proposed using object-based frame interpolation schemes that need a segmentation map as overhead information [4]. Kawaguchi [5] exploited neighborhoods of block-based motion vectors and the displacement frame difference to detect multiple motion inside block. Our approach is different from Kawaguchi's in that we take care of occlusion explicitly. The proposed algorithm needs to perform separate motion estimation only on a few of selected pixels.

The main contribution of this paper is to exploit a coarse blocky motion vector field to obtain a dense motion vector field through mesh-based mapping while providing viable

0-7803-6725-1/01/$10.00 ©2001 IEEE

interpolation results. We also provide a method for analyzing relative depth cue for interpolation. This paper is organized as follows: First, the problems of mesh-based motion estimation are discussed. Second, the relative depth cue at motion boundaries is analyzed. Third, a technique for adaptive interpolation filtering is proposed. Finally, experimental results are presented and conclusions are shared.

2. PROBLEMS OF MESH-BASED MOTION ESTIMATION

In the decoder we wish to interpolate any missing frames between two temporally adjacent frames. For the purpose of this work, we decompose a video frame at time t into an array of non-overlapped blocks and then assign a motion vector to each block. We assume that the motion vector can be available from the compressed bit stream or estimated in the decoder as post-processing. Unlike conventional frame interpolation methods in which missing intensity values are interpolated using the given blockwise constant vector field or the re-estimated dense motion vector field, we use the block-based motion vectors. We refine the block-based motion vectors by estimating motion only on a small fraction of pre-defined sets of pixels. We define a set of pixels according to a triangular mesh topology. To do this, we sample pixels uniformly within a certain area and then enroll them as nodes for triangulation of the frame at time t. Because the mesh topology is fixed, we can decompose the frame into non-overlapping uniform triangles.

2.1. Change detection and mesh generation

For motion segmentation, we use a robust change detection mask that is obtained from the frame difference between the given two frames. We set a threshold value to quantize the frame difference image into two states: changed or unchanged. After detecting the changed pixels, we apply a 2-D median filter to remove isolated pixels. Then we use a 2-D morphological filter to remove the isolated pixels and to obtain reliably connected pixels. We finally apply a size filter to the morphologically filtered image to remove any small size of changed pixels below a threshold. Fig. 1 shows an example obtainable from the aforementioned procedure. We can observe from the figure that the resulting changed region has no unchanged pixels inside and also has motion boundaries and disoccluded pixels around its boundary. Based on this observation, we assume that the changed region consists of the moving foreground and the static background. Now, the uniform mesh confined by the proposed topology is designed only within a box that encloses the changed region. We assume that a pixel belonging to the unchanged region at time t has a correspondence at the same location at time t-1. Then, it is straightforward to interpolate the intensity values at any time between time t-1 and t. Next, we will discuss how to improve the accuracy of uniform mesh-based motion estimation with the aid of a block-based motion vector field and a pel-recursive motion estimator.

2.2. Mesh-based motion estimation

We use a pel-recursive motion estimation algorithm for nodal displacement estimation. A typical pel-recursive motion estimation algorithm solves a deterministic optimization problem by iteratively searching the optimal displacement at the current pixel such that the motion vector minimizes the motion compensation error. The usual practice is to assume that the initial displacement is zero. In our case, we can exploit a *non-causal neighborhood* of the current node. In fact, we can specify the initial condition from the nine nearest neighbors whose motion vectors are already determined by previous processing, looking in a 3×3 window of block-based motion vectors containing the current block's vector at its center. To this effect, we can avoid the local minima at which the solution of the optimization problem might be trapped. Note that pel-recursion leads to fractional pixel accuracy motion vectors. For computational simplicity, we round the fractional numbers to half-pixel accuracy.

Six affine transform parameters can be obtained from the estimated three nodal displacement vectors at each triangle. Because we know the displacement vector at each node, we can calculate the affine parameters uniquely. We assume that motion of a rigid planar surface provides affine transformation under the orthographic projection. It is also valid to assume that triangulation on the rigid planar surface yields a smooth motion vector field. For simplicity, we assume that the changed region mentioned in Section 2.1 can be modeled by a planar surface. We confine the movement of each vertex to guarantee the smoothness of the motion vector field. For this goal, we check the topology of mesh at a node during pel-recursion. A neighborhood of a node is defined as a polygon made of all triangles whose vertices are connected to the node. To prevent the motion of the node from breaking out of the neighborhood, we restrict our pel-recursion within the polygon. Note that this smoothness constraint does not allow any rip on the triangulation and thus it is not suitable for the analysis of object/motion boundaries where motion estimation produces large estimation errors.

3. ANALYSIS OF INVALID TRIANGLES

For analysis boundaries can be represented with strong edges. The model used here is a mixture of a Hough transform and a Sobel operator.

We first find the triangles that violate a pre-defined requirement. The requirement is based on the measurement of the motion compensation errors. Motion compensation can be accomplished either by an inverse affine transform or by an inverse texture mapping. In this paper, we use the inverse texture mapping which is faster than the inverse affine transform. After calculating the average of the motion compensation errors at each triangle, we compare the average with a pre-set value and then decide whether the motion of the triangle is valid or not. If the average error is unacceptably large, we assume that the motion vector field of the triangle cannot be used for the interpolation. We call such a triangle an invalid triangle. When we closely examine invalid triangles, we observe that they exist most probably on the boundaries. This observation can be explained from the viewpoint of the topology of the triangulation and the smoothness constraint. If a node is near the motion boundary, the motion estimation results for nodes in its neighborhood depend on their positions relative to it. For example, if the node is on the background, the neighborhood triangles whose other two vertices are on the foreground will produce wrong affine parameters. Note that all three vertices of a triangle must be on the same object surface due to the smoothness constraint. An example of invalid triangles is shown in Fig. 2. Invalid triangles can be modeled in order to deal with the boundaries. We assume that the boundaries are approximated by strong edges. Once strong edges approximate the boundaries, the remaining task is to segment pixels into the moving foreground and the static background. To accomplish this goal, we find edges within each invalid triangle by using a 3×3 Sobel operator. Linear filters are applied to the edges in order to remove both isolated and weak edges. Then we segment each invalid triangle into the two probable regions. A Hough transform identifies the best possible line from the edge points. In doing so, we bisect each invalid triangle.

Finally, we re-estimate the motion of invalid triangles. If a segmented region belongs to the moving foreground, its motion can be predicted from the affine motion parameters of neighboring triangles. We define the neighboring triangles as triangles connected to a vertex (or two vertices) belonging to the segmented region of the invalid triangle. The motion parameters of the neighboring triangles must be valid. The prediction is done by a brute-force search of the best matched set of affine parameters among the candidates. The remaining region is assumed to belong to the appearing background. The motion of the region can be obtained through a similar procedure.

4. ADAPTIVE INTERPOLATION FILTERING

The non-linearity of the motion field, i.e., the discontinuity, leads to adaptive interpolation. After analyzing the invalid triangles, we have four types of regions: the unchanged region, the valid triangles, the moving regions of the invalid triangles, and the appearing background of the invalid triangles. Each region has its own motion vectors. We use a different interpolation filter according to the region type. A zeroth order linear interpolation filter is applied to the missing pixels belonging to the unchanged region. A bilinear interpolation filter is applied to the missing pixels between the points sitting at the end of the motion trajectory obtained from the valid triangles. The same filter is applied to the moving regions of the invalid triangles. The pixels belonging to the background are interpolated first. Then the pixels belonging to the foreground are interpolated. This procedure simply solves the interpolation problem of the appearing background due to our knowledge of the relative depth.

5. RESULTS

We implemented the proposed algorithm using various QCIF (176x144) video sequences. Fig. 3.(a) shows a typical motion vector field resulting from the block matching algorithm (BMA). The block size for the BMA is set to 16x16. We tried to make the number of triangles of a uniform mesh the same as the case of the BMA. After fixing the locations of node points, the algorithm obtains a motion vector per node in order to estimate affine parameters per triangle. Fig. 3.(b) shows the improved motion vector field. Fig. 4 shows comparisons among the block-based method, the pel-recursive method, and the proposed method. The frame numbers in Fig. 4 refer to the interpolated frames. It means that we do not include the PSNRs of the transmitted frames in the graphs. The proposed algorithm shows a performance close to the pel-recursive approach in terms of PSNR. Fig. 5 shows a comparison between two interpolated images using the block-based method and the proposed method, respectively.

6. CONCLUSIONS

We propose a new frame interpolation algorithm for video sequences compressed at very low bandwidths. We represent a model of motion boundaries using a mesh topology and motion compensation errors. Pixelwise motion vectors are obtained from the transmitted motion vectors. This algorithm improves the performance of interpolation, especially around motion boundaries. The proposed algorithm can be adopted for real-time implementation with code optimization.

7. REFERENCES

[1] J. Biemond, L. Looijenga, and D. E. Boekee, "A pel-recursive wiener-based displacement estimation algo-

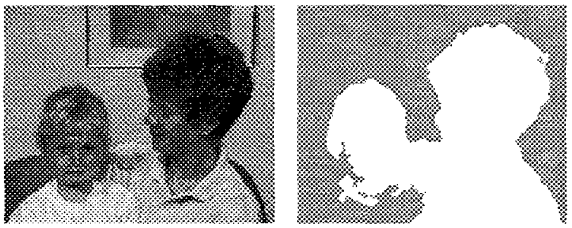
rithm for interframe image coding applications," *SPIE Visual Communications and Image Processing II*, vol. 845, pp. 424–31, 1987.

[2] Ciro Cafforio, Fabio Rocca, and Stefano Tubaro, "Motion compensated image interpolation," *IEEE Trans. on Communications*, vol. 38, no. 2, pp. 215–222, Feb. 1990.

[3] Fernando C. M. Martins, "Real-time video frame rate adaptation based on warping of edge-preserving meshes," *IEEE International Conference on Image Processing*, pp. 948–952, 1999.

[4] Soo-Chul Han and J. W. Woods, "Frame-rate up-conversion using transmitted motion and segmentation fields for very low bit-rate video coding," *IEEE International Conference on Image Processing*, pp. 747–750, 1997.

[5] Kunio Kawaguchi and Sanjit K. Mitra, "Frame rate up-conversion considering multiple motion," *IEEE International Conference on Image Processing*, pp. 727–730, 1997.



(a) Original (b) Change Detection

Fig. 1. Example of change detection: frame 5 of the mother & daughter sequence

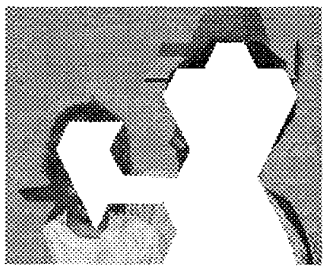
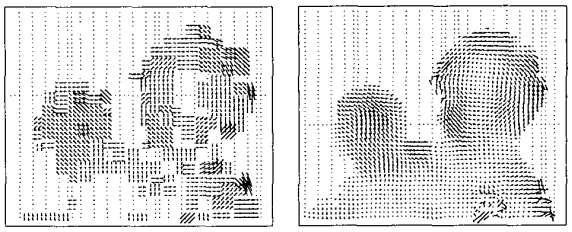
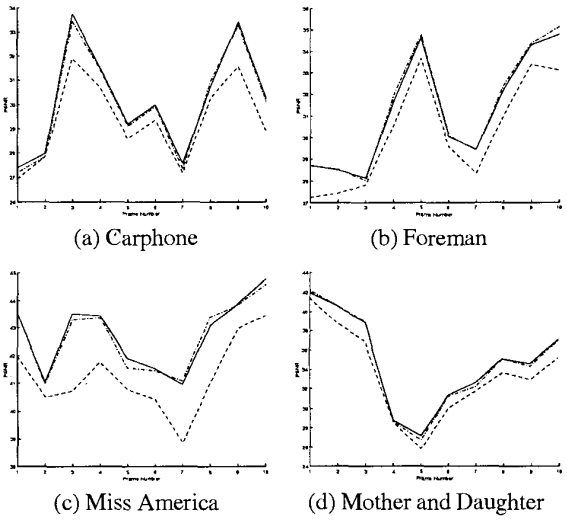


Fig. 2. Example of invalid triangles: The white pixels belong to valid triangles. The gray pixels belong to the unchanged region. The pixels that remain around the boundaries belong to invalid triangles.



(a) Block-based MV field (b) Improved MV field

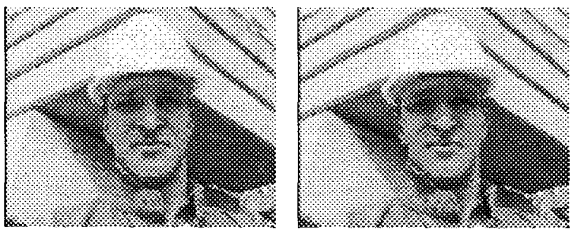
Fig. 3. Example of motion vector (MV) fields



(a) Carphone (b) Foreman

(c) Miss America (d) Mother and Daughter

Fig. 4. Comparison of PSNR: block-based method ('–': dashed line), pel-recursion method ('-': solid line), and proposed method ('---': dashed line with dots)



(a) block-based interpolation (a) proposed interpolation

Fig. 5. Comparison of interpolated images