MOTION COMPENSATED PREDICTION USING PARTIAL MESH GENERATION

Han Huang[a,b,c], John W. Woods[b] and Yao Zhao[a,c]

[a]Institute of Information Science, Beijing Jiaotong University, Beijing, China
[b]ECSE Department Rensselaer Polytechnic Institute, Troy, NY, USA
[c]Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing, China

ABSTRACT

In this paper, a new motion modeling method is introduced. We employ the popular quadtree structure to divide an image frame into variable size blocks. Each block is considered as an independent deformable mesh cell, and can be connected to causal neighbor cells to produce partial meshes. The motion model in each mesh cell is adaptively selected by minimizing a Lagrangian cost. We evaluate the proposed method by motion compensated prediction and coding. Preliminary experimental results show its potential advantages.

Index Terms— Polynomial motion, affine, mesh, merging, motion compensation, quadtree

1. INTRODUCTION

Polynomial motion models have advantages over translational motion models in capturing more complex motion. The mesh-based approach [1] produces a continuous motion field, thus visually more pleasing motion-compensated prediction frames. However, the non-causal spatial dependence among the control points makes it difficult for rate-distortion optimization. Also, it cannot capture the motion discontinuity at the boundary of moving objects. Alternatively, a polynomial motion model can be adopted within the traditional block matching framework, with the penalty of increased motion parameters. In [2], an orthogonalization scheme along with motion assisted merging and motion-model adaptation was developed to efficiently code the polynomial parameters. Mathew and Taubman investigated a polynomial motion model within the framework of quadtree motion modeling with leaf merging [3, 4]. In the aforementioned non-mesh based approaches, motion is represented by estimated polynomial parameters instead of motion vectors at control points.

In this paper, a similar quadtree decomposition and bottom-up pruning framework is adopted. Each node is considered as an independent mesh cell. Various motion modes are defined according to different relations among the motion vectors at the control points. For each mesh cell, the best motion model is selected by a rate-distortion optimization. Furthermore, the mesh cell is allowed to connect to its causal neighbors, resulting in a partial mesh with a continuous motion field. The connection scheme is similar to the merging technique of [3, 4]. However, connection of mesh cells discussed in this paper is applied on control points, which does not force a single motion model on the entire merged region. In other words, each cell may still undergo a separate motion after connection, providing more flexibility.

2. MOTION MODELING

The quadtree structure is adopted to recursively divide a motion block into smaller blocks. Motion in each block is modeled by an independent mesh cell with 4 control points at the corners. Consider a mesh cell with size $s \times s$. Let (x_i, y_i) and $\vec{v}_i = (v_{x_i}, v_{y_i})$ be the coordinate and motion vector of control point $i, i = 1, 2, 3, 4$. Since the affine motion model is chosen in this paper, each mesh cell is treated as a pair of triangles. The partition can be either *LU* or *UL* as shown in Fig. 1. The corresponding motion modes are named *LU_AFFINE* and *UL_AFFINE* respectively. Note that the two triangles share two control points. Let (x, y) be a point in the current frame, and (x', y') be the corresponding point in a reference frame. The affine transform is described as

$$\begin{cases} x' = ax + by + e \\ y' = cx + dy + f \end{cases}. \tag{1}$$

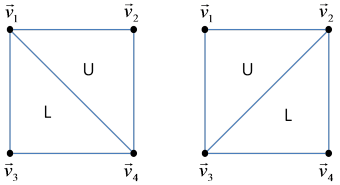
Here a, b, c, d, e and f are the affine parameters. Define the motion at (x, y) as $(v_x, v_y) = (x - x', y - y')$, it then follows that

$$\begin{cases} v_x = (1-a)x - by - e \\ v_y = (1-c)x - dy - f \end{cases}. \tag{2}$$

Motion-compensated prediction is seen as $\hat{I}_t(x, y) = \tilde{I}_{t-1}(x', y')$, where $\hat{I}_t$ is the predicted frame and $\tilde{I}_{t-1}$ is the interpolated reference frame using cubic convolution interpolation [5].

Transform the coordinate system by setting $(x_1, y_1) = (0, 0)$. Substituting (x', y') and (x, y) with (x'_i, y'_i) and (x_i, y_i), $i = 1, 3, 4$ in equation (1) and solving the resulting equations, we can derive the affine transform parameters for (x, y) in the lower triangle of the *LU* partition.

$$s \begin{bmatrix} a & b & e \\ c & d & f \end{bmatrix} = \begin{bmatrix} (v_{x_3} - v_{x_4}) + s & v_{x_1} - v_{x_3} & -v_{x_1} s \\ (v_{y_3} - v_{y_4}) + s & v_{y_1} - v_{y_3} & -v_{y_1} s \end{bmatrix}. \tag{3}$$



(a) LU partition (b) UL partition

Fig. 1: Mesh Cells

Then, by equation (2), we have

$$\begin{cases} sv_x = (s-y)v_{x_1} + (y-x)v_{x_3} + xv_{x_4} \\ sv_y = (s-y)v_{y_1} + (y-x)v_{y_3} + xv_{y_4} \end{cases}. \quad (4)$$

Similarly, using control points 1, 2, and 4 we can obtain

$$\begin{cases} sv_x = (s-x)v_{x_1} + (x-y)v_{x_2} + yv_{x_4} \\ sv_y = (s-x)v_{y_1} + (x-y)v_{y_2} + yv_{y_4} \end{cases}, \quad (5)$$

for (x, y) in the upper triangle. Through the same derivation, similar equations follow for the *UL* partition. These equations show the affine motion as a linear interpolation between the motion vectors at control points.

Define $\vec{v}_0$ as the translational motion of a mesh cell. It's used as the search center for $\vec{v}_i$, i.e. $\vec{v}_i = \vec{v}_0 + \Delta\vec{v}_i$. The motion vectors that need to be transmitted are: $\vec{v}_0$ and $\Delta\vec{v}_i, i = 1, 2, 3, 4$. Obviously, transmitting five motion vectors, in place of the one needed for block matching, will increase the motion rate dramatically. And it's not necessary to use higher order motion for every block. So we define 3 additional motion modes as follows.

- Horizontal Bilinear: set $\vec{v}_3 = \vec{v}_1$ and $\vec{v}_4 = \vec{v}_2$. This motion field is uniform within each column of the block, and is bilinearly interpolated in the horizontal direction. The motion vectors that need to be transmitted are: $\vec{v}_0$, $\Delta\vec{v}_1$ and $\Delta\vec{v}_2$. We refer to this mode as *HOR*.
- Vertical Bilinear: set $\vec{v}_2 = \vec{v}_1$ and $\vec{v}_4 = \vec{v}_3$. This motion field is uniform within each row of the block, and is bilinearly interpolated in the vertical direction. The motion vectors that need to be transmitted are: $\vec{v}_0$, $\Delta\vec{v}_1$ and $\Delta\vec{v}_3$. We refer to this mode as *VER*.
- Translational: set $\vec{v}_i = \vec{v}_0, i = 1, 2, 3, 4$. This motion field within the block is uniform, i.e. translational motion as in conventional block matching. Only $\vec{v}_0$ needs to be transmitted. We refer to this mode as *TRANS*.

Note that *HOR*, *VER* and *TRANS* modes are special cases of affine motion.

3. MESH CELL CONNECTION

The connection of mesh cells is actually merging of adjacent control points. We want to favor the advantages of a mesh based approach by generating a partial mesh covering an arbitrary region, and at the same time further reduce the number of motion vectors. A neighbor cell is said to be valid for connection if it *contains* the boundary edge of the current cell. We illustrate this in Fig. 2, where A is valid for connection in (a) and (b) but not in case (c). There are four possible connections: *left_connected*, *up_connected*, *left_up_connected* and *un_connected*. By *left_up_connected* we mean that the current mesh cell is connected to both its left and upper neighbors. As shown in Fig. 2d, B is connected to C, A is also connected to C, A and B are connected by one control point, then X is able to connected to both A and B.

When the current cell X is connected to its neighbors, the motion vectors at the connected control points and $\vec{v}_0$ are inferred by those of its neighbors. For example, in the case of Fig. 2a, $\vec{v}_0^X = \vec{v}_0^A, \vec{v}_1^X = \vec{v}_2^A, \vec{v}_3^X = \vec{v}_4^A$. In Fig. 2b when two mesh cells have different sizes, $\vec{v}_3^X$ is obtained by bilinear interpolation of $\vec{v}_2^A$ and $\vec{v}_4^A$. Setting $\vec{v}_0^X = \vec{v}_0^A$ is based on the intuition that the connected mesh cells may belong to the same object, and thus have a similar translational motion. For simplicity, no previous modes are changed when a connection is made. So *HOR*, *VER* and *TRANS* mesh cells may be connected to prior neighbors that do not honor their constraints on $\vec{v}_i$. Then the constraints defined in the 3 modes are only applied on the opposite face. Take Fig. 2a for example, if X is connected to A, then *HOR* means only setting $\vec{v}_4^X = \vec{v}_2^X$ and not changing $\vec{v}_1^X$ and $\vec{v}_3^X$ which are inferred by $\vec{v}_2^A$ and $\vec{v}_4^A$. Then since affine motion is used, X must be split, either *LU* or *UL*. In this case, *LU* partition is always assumed in our present implementation. Note that all motion modes are equivalent to linear interpolation between control points. The resulting motion field is then continuous across the boundary of two connected mesh cells, even if they have different modes.

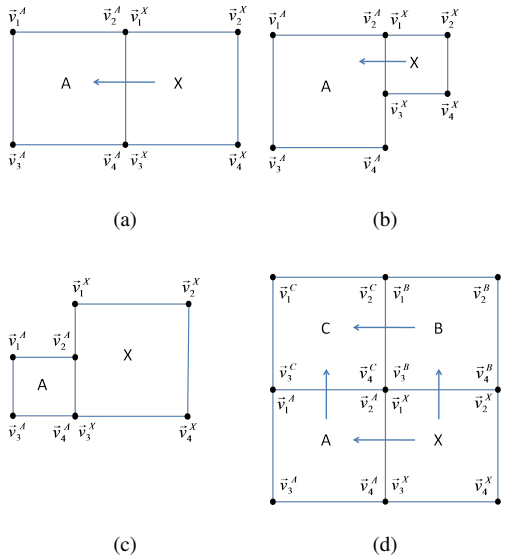
4. ESTIMATION AND CODING OF MOTION PARAMETERS

Motion parameter estimation is conducted within the quadtree decomposition and bottom-up pruning procedure [6]. Let J_0 be the Lagrangian cost of a mesh cell, and $J_k, k = 1, 2, 3, 4$ be the costs of its four child mesh cells. The children are pruned if $J_0 \leq \sum_{k=1}^{4} J_k$. In the rest of this section, we will describe the procedure for a given mesh cell.

Let D be the prediction error and R be the estimated coding bits. Denote Φ as the set of connection options described in Section 3 and Ψ as the set of motion modes described in Section 2. Let $\mathbf{c} \in \Phi$ and $\mathbf{m} \in \Psi$ denote connection option and motion mode. Then we exhaustively search in the space $(\Phi \times \Psi)$ to find an optimal $(\mathbf{c}^*, \mathbf{m}^*)$ with a minimum J^*. For each $(\mathbf{c}, \mathbf{m})$, $J(\mathbf{c}, \mathbf{m})$ is obtained by

$$\arg\min_{\mathbf{v}} J(\mathbf{c}, \mathbf{m}, \mathbf{v}) = D(\mathbf{c}, \mathbf{m}, \mathbf{v}) + \lambda R(\mathbf{c}, \mathbf{m}, \mathbf{v}) \quad (6)$$

where λ is a Lagrange multiplier, $\mathbf{v} \subset \{\vec{v}_0, \Delta\vec{v}_i, i = 1, \ldots, 4\}$



(a) (b) (c) (d)

Fig. 2: Connection examples

is a set of *free* motion vectors [1] depending on $(\mathbf{c}, \mathbf{m})$. Searching on $\mathbf{v}$ is described as follows.

1. If $\vec{v}_0$ is a *free* motion vector, find $\vec{v}_0$ by minimizing $J(\vec{v}_0) = D(\vec{v}_0) + \lambda R(\vec{v}_0 - \vec{v}_p)$. where $\vec{v}_p$ is a median predictor, obtained from $\vec{v}_0$ of neighbor mesh cells. Here we may employ a SKIP mode for $\vec{v}_0$ when $\mathbf{c}$ is *un_connected*. If SKIP is chosen, then $\vec{v}_0 = \vec{v}_p$ and $R(\vec{v}_0 - \vec{v}_p) = 0$.

2. For $i = 1, 2, 3, 4$, if $\vec{v}_i$ is *free* motion vector, find $\Delta\vec{v}_i$ by minimizing $J(\Delta\vec{v}_i) = D(\vec{v}_0 + \Delta\vec{v}_i) + \lambda R(\Delta\vec{v}_i)$ while keeping other motion vectors fixed.

3. If there is more than one *free* control point[2], step 2 is iterated until no motion vectors are changed or a predefined maximum iteration number 3 is achieved.

We use a binary flag, denoted as *connected_sign*, to indicate whether the current cell is connected or not, and a two-bit codeword, denoted as *connection_direction*, is used to represent *left_connected*, *up_connected* or *left_up_connected*. Let n be the number of valid neighbor cells for connection. If $n > 0$, then *connected_sign* is signaled to the decoder. If $n > 1$ and a connection decision is made, *connection_direction* is further sent. For coding of mode selection, we also use a binary flag to indicate whether the mode is *TRANS*. If not, another two-bit codeword is used for representing the other 4 modes. In (6), $\mathbf{c}$ is denoted by 1 bit for *connected_sign* and 2 bits for *connection_direction*. The mode $\mathbf{m}$ is denoted by 1 bit for *TRANS* mode and 2 additional bits for the other modes. An exp-Golomb coder is used to estimate the coding rate of $\mathbf{v}$ in the optimization, but in the actual coding, $\mathbf{c}$ and $\mathbf{m}$ are coded by binary arithmetic coding and $\mathbf{v}$ is coded by the entropy coding extension of CABAC presented in [7].

[1] a motion vector that is not inferred by a neighbor cell or constrained by the definition of the motion mode

[2] control point with free motion vector

5. EXPERIMENTAL RESULTS

We evaluate the proposed method by motion-compensated predictive or hybrid coding. We will refer to the proposed method as Progressive Mesh Generation(*ProMesh*), and the method without connection as Generalized Variable Size Block Matching(*GVSBM*). We also implement Variable Size Block Matching(*VSBM*), using the same quadtree setting and coding procedure, for comparison purposes. The original reference (previous) frame is used to predict the current frame. The block size is varied from 32×32 to 4×4. The search range is $[-31, 31]$ for $\vec{v}_0$, and $[-3, 3]$ for $\Delta\vec{v}_i, i = 1, 2, 3, 4$. The motion vectors have quarter-pixel accuracy. Plots of motion rate versus prediction PSNR were obtained by varying the λ value, for the first 100 frames of *FOREMAN* and 150 frames of *BUS*, and are shown in Figs. 3 and 4, respectively. Both test clips are CIF @ 30 fps. In the range of middle to high bitrate, *GVSBM* outperforms *VSBM* by $0.3 \sim 0.6$ dB for *FOREMAN* and $0.3 \sim 1$ dB for *BUS*. The *ProMesh* method further improves the performance by $0.2 \sim 0.3$ dB. There's little gain at low bitrates, since the overhead of transmitting more motion information becomes significant. The *general_spt_no_merge* and *general_spt* from Fig. 2 of [3] are added in Fig. 3 as reference. The reader should note that SKIP mode was not implemented in [3], and this may account for some of the difference. But our experiments show that the SKIP mode only improves performance by about 0.1 dB for any of the three methods tested here. For actual coding, we code the first frame as I frame and all the other frames as P frames. The EZBC image coder [8] is used to code all frames. Both luma and chroma are coded and counted in the bitrate. No intra-prediction, overlapped block motion compensation or de-blocking filter is used. The bitrate is constant over all P frames. And the rate of I frame is 6 times of P frame rate. The λ value was optimized for *VSBM*, then the same value was used for *GVSBM* and *ProMesh*. Given a fixed rate, a gain of $0.52 \sim 0.72$ dB is observed for *FOREMAN*, and $0.26 \sim 0.56$ dB for *BUS*. Equivalently, it's $15 \sim 20$ percent bitrate saving on average. Due to the limited space, we only plot the Y-PSNR curve of *FOREMAN* in Fig. 5. The motion-compensation predicted and coded/decoded sequences are found at [9] posted in .yuv format. We can observe that the sequences produced by the proposed method are visually more pleasing.

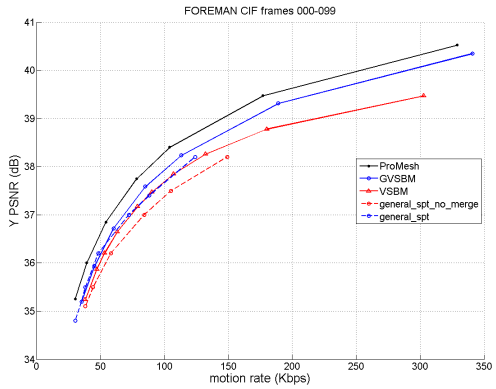


Fig. 3: motion-compensated prediction performance *foreman*

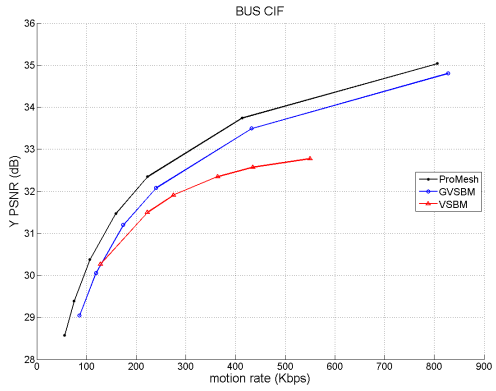


Fig. 4: motion-compensated prediction performance *bus*

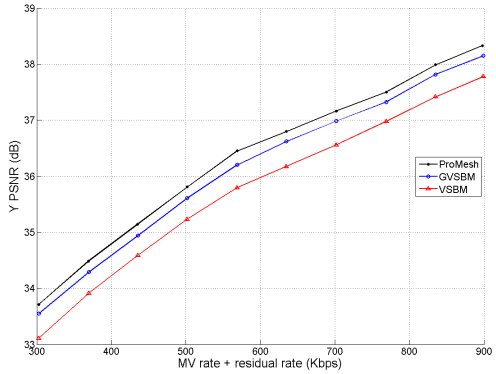


Fig. 5: Coding results *foreman*

6. CONCLUSION

In this paper, a new adaptive polynomial motion model is proposed for motion-compensated prediction and hybrid coding. We described the motion modeling by considering each block as a mesh cell. Then a connection scheme was adopted to progressively connect those mesh cells into partial meshes. Although the proposed *ProMesh* scheme is a greedy method, some preliminary experimental results have shown its advantages. It's evident that a second pass refinement of the motion vectors of connected control points can further improve the performance.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61025013, No. 60972085) , and Sino-Singapore JRP (No. 2010DFA11010).

7. REFERENCES

[1] Y. Nakaya and H. Harashima, "An iterative motion estimation method using triangular patches for motion compensation," in *Visual Communication and Image Processing*, 1991, vol. 1605, pp. 546–557.

[2] M. Karczewicz, J. Niewgowski, and P. Haavisto, "Video coding using motion compensation with polynomial motion vector fields," *Signal Processing: Image Communication*, vol. 10, pp. 63–91, 1997.

[3] R. Mathew and D. S. Taubman, "Hierarchical and polynomial motion modeling with quad-tree leaf merging," in *IEEE International Conference on Image Processing*. IEEE, 2006, pp. 1881–1884.

[4] R. Mathew and D. S. Taubman, "Quad-tree motion modeling with leaf merging," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 10, pp. 1331–1345, 2010.

[5] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 6, pp. 1153–1160, 1981.

[6] G.J. Sullivan and RL Baker, "Efficient quadtree coding of images and video," *IEEE Trans. Image Process.*, vol. 3, no. 3, pp. 327–331, 2002.

[7] Yongjun Wu, *Fully scalable subband/wavelet video coding system*, Ph.D. thesis, Rensselaer Polytechnic Institute, 2005.

[8] S.T. Hsiang, *Highly scalable subband/wavelet image and video coding*, Ph.D. thesis, Rensselaer Polytechnic Institute, 2002.

[9] "www DOT cipr DOT rpi DOT edu/~huangh7/icip2011," [Online].