Control-Point Representation and Differential Coding Affine-Motion Compensation

Han Huang, *Student Member, IEEE,* John W. Woods, *Fellow, IEEE,* Yao Zhao, *Senior Member, IEEE,* and Huihui Bai

***Abstract*—The affine-motion model is able to capture rotation, zooming, and the deformation of moving objects, thereby providing a better motion-compensated prediction. However, it is not widely used due to difficulty in both estimation and efficient coding of its motion parameters. To alleviate this problem, a new control-point representation that favors differential coding is proposed for efficient compression of affine parameters. By exploiting the spatial correlation between adjacent coding blocks, motion vectors at control points can be predicted and thus efficiently coded, leading to overall improved performance. To evaluate the proposed method, four new affine prediction modes are designed and embedded into the high-efficiency video coding test model HM1.0. The encoder adaptively chooses whether to use the new affine mode in an operational rate-distortion optimization. Bitrate savings up to 33.82% in low-delay and 23.90% in random-access test conditions are obtained for low-complexity encoder settings. For high-efficiency settings, bitrate savings up to 14.26% and 4.89% for these two modes are observed.**

***Index Terms*—Affine-motion model, high-efficiency video coding (HEVC), motion-compensated prediction (MCP), motion estimation, video coding.**

I. Introduction

In state-of-the-art video coders, motion-compensated prediction (MCP) is very important to obtain high coding efficiency [1]. In conventional MCP methods, the block matching algorithm is widely adopted. As shown in Fig. 1(a), a best matching block in a reference frame, after shifting by the motion vector (MV) $\vec{v}$, is used to predict the current block. Block matching assumes that the motion in a given block is uniform, i.e., employs a translational motion model. Thus, it fails to capture rotation, scaling, and other deformations of moving objects. To solve this problem, higher order motion models were introduced in video coding.

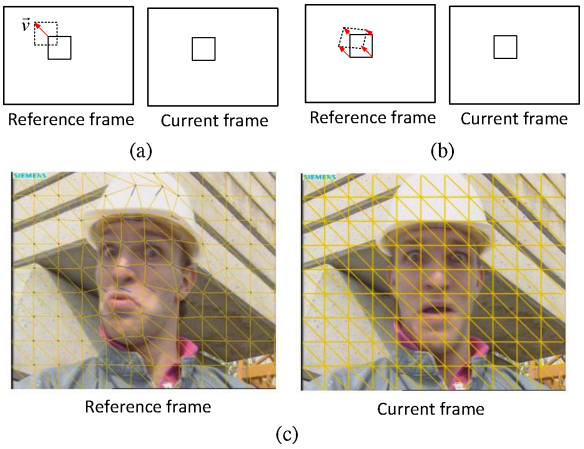


Fig. 1. (a) Block matching. (b) Generalized block matching. (c) Mesh matching [2].

Approaches to higher order motion models include mesh matching [3], [4] and control grid interpolation [5]. In these mesh-based methods, the motion is represented by displacements of the so-called control points. Then, the image is warped accordingly, producing a visually more pleasing MCP frame than does block matching. An example of mesh matching is shown in Fig. 1(c). Since a control point is shared by its surrounding patches, the overhead of transmitting the motion information is generally similar to that of fixed-sized block matching. However, the noncausal spatial dependence among the control points makes operational rate-distortion optimization difficult, especially for variable size adaptation. In [6] and [7], empirical thresholds were used for splitting. Other mesh-based techniques include the closed-form solution [8] and the content-based mesh [9], [10]. In another approach, higher order motion models are adopted by generalizing block matching [11], [12], as shown in Fig. 1(b), allowing reference-frame blocks to be deformed for better prediction but with the penalty of increased motion parameters per block. In [12], the authors proposed an orthonormalization scheme that makes the motion parameters less sensitive to quantization. Motion-assisted merging and motion-model adaptation were also developed to compress the polynomial parameters. Other block-based approaches can be found in [13]–[17].

While not used in standard-based video coders, higher order motion models have been found useful in research codecs. In

Manuscript received October 04, 2012; revised December 19, 2012; accepted February 08, 2013. Date of publication March 27, 2013; date of current version September 28, 2013. This work was supported in part by the 973 Program under Grant 2012CB316400; the National Natural Science Funds for Distinguished Young Scholar under Grant 61025013; the National Natural Science Foundation of China, under Grant 61210006, 61202240, 60903066, 61272051, and 6101393; JPNSF(BK2011455), and the Program for Changjiang Scholars and Innovative Research Team in University. This paper was recommended by Associate Editor L. Zhang.

H. Huang, Y. Zhao, and H. Bai are with the Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China (e-mail: huanghan@bjtu.edu.cn; yzhao@bjtu.edu.cn; hhbai@bjtu.edu.cn).

J. W. Woods is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA 12180-3590 USA (e-mail: woods@ecse.rpi.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2013.2254977

[18], a global affine-motion model was introduced to generate better reference frames. A 4-D vector quantizer was developed in [19] to code the scaling parameters in the multiple global affine-motion model. In [20], a single global affine transform was adopted based on image alignment. In [21], a feature-based robust global-motion method was used to estimate the homography transform between the current and reference frames. Then, a parametric SKIP mode was designed based on the estimated global motion. In [22], the authors employed an in-loop postprocessing method benefiting from the advantages of affine-motion prediction and using conventional block MVs. A preprocessing method was proposed in [23]. Muhit *et al.* [24] proposed an elastic motion model that allows larger blocks to be used. Other researchers focused their work on zoom motion [25]–[28]. In our previous work [29], a coding block was regarded as a part of a mesh grid consisting of two triangular patches. A partial mesh connection concept was adopted to reduce the number of MVs. In [30], we introduced affine motion into SKIP/DIRECT prediction modes and demonstrated its efficiency. In this paper, the idea is generalized to a new control-point representation of affine motion favoring efficient differential motion-vector coding.

This paper is organized as follows. The proposed block-based affine-motion model is presented and discussed in Section II. Followed by implementations in Section III, four new affine modes are designed and embedded into the high-efficiency video coding (HEVC) test model HM1.0. In Section IV, numerous experimental results are shown and discussed. Bitrate savings compared to the original HM1.0 coder are up to 33.82% in the low-delay IPPP mode and 23.90% in the random-access hierarchical B mode under low-complexity encoder settings. For high-efficiency encoder settings, bitrate savings are up to 14.26% and 4.89% for these two modes, respectively. Finally, conclusions are drawn in Section V.

II. Proposed Block-Based Affine-Motion Model and Differential Coding Method

The 2-D affine transform is described as

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

where (x, y) and (x', y') are a pair of corresponding locations in the current and reference frames, respectively, and a, b, c, d, e, and f are the affine parameters. Let $(v_x, v_y) = (x - x', y - y')$ be the apparent motion at location (x, y) in the current frame; it then follows that

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

which is called the affine-motion model. In contrast to the conventional block matching algorithm, the matching block in the reference frame is allowed to be warped or deformed for better MCP. The cost is the transmission of more motion parameters, six instead of two per block. In video coding, the motion parameters are transmitted as side information that can be significant at low total bitrates. It has thus been found that the compression of this motion information is essential to improve overall coding performance. The resulting increase in the number of motion parameters could neutralize the advantages of affine-motion compensation. Therefore, efficient coding of motion parameters becomes more critical for the affine-motion model.

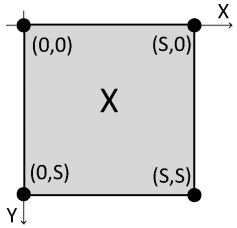


Fig. 2. Control-point representation of a square region.

Direct coding of the polynomial coefficients in (1) requires a special quantizer. In [12], the authors proposed to orthonormalize the affine parameters since they are sensitive to quantization. In [31], the affine coefficients were examined by deriving a quadratic relationship between the MCP residual signal and the quantization step size. In [15], the affine parameters were coded as three symbols: the nonzero pattern, the amplitudes, and the signs. For efficient encoding, the probability distributions of these symbols were addressed and various context types were designed to exploit correlations between these parameters. Alternatively, the affine-motion parameters can be represented by displacements at control points as translational MVs. In this way, the existing motion-vector coding method can be employed. This also solves the problem of MV prediction across different motion models since it is achieved by predicting the translational motion at the designated locations of the control points. In the following section, the proposed control-point representation is described and its advantages in differential motion-vector coding are discussed.

A. Proposed Control-Point Representation

Given a square region $\mathcal{X}$ of size $S \times S$, set the coordinate system as shown in Fig. 2. The top and left three corners, i.e., $(0, 0), (S, 0)$, and $(0, S)$, can serve as control points, denoted as $(x_i, y_i), i = 0, 1, 2$. Let their translational MVs be $\vec{v}_i = (v_{x_i}, v_{y_i}), i = 0, 1, 2$. The matrix $\mathbf{V} \triangleq (\vec{v}_0, \vec{v}_1, \vec{v}_2)$ will be referred to as the affine-motion matrix. The displaced control points that deform $\mathcal{X}$ are $(x_i', y_i') = (x_i - v_{x_i}, y_i - v_{y_i})$. By substituting (v_x, v_y) and (x, y) into (2) with (v_{x_i}, v_{y_i}) and (x_i, y_i), we will have six equations with six unknowns a, b, c, d, e, and f. Solving these equations, we have

$$\begin{cases} a = 1 - \dfrac{v_{x_1} - v_{x_0}}{S}, b = \dfrac{v_{x_0} - v_{x_2}}{S}, e = -v_{x_0} \\ c = 1 - \dfrac{v_{y_1} - v_{y_0}}{S}, d = \dfrac{v_{y_0} - v_{y_2}}{S}, f = -v_{y_0}. \end{cases} \tag{3}$$

By (2) and (3), we get

$$\begin{cases} v_x = \dfrac{v_{x_1} - v_{x_0}}{S}x + \dfrac{v_{x_2} - v_{x_0}}{S}y + v_{x_0} \\ v_y = \dfrac{v_{y_1} - v_{y_0}}{S}x + \dfrac{v_{y_2} - v_{y_0}}{S}y + v_{y_0}. \end{cases} \tag{4}$$

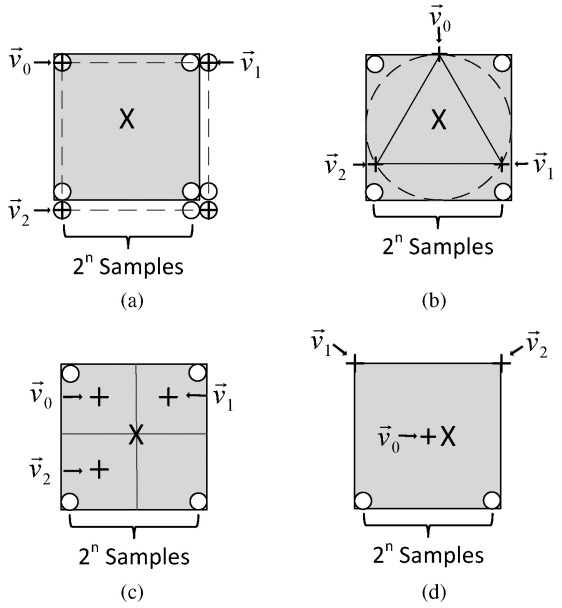


Fig. 3. Comparison of control-point representations: our method and others in the literature. Black circles indicate pixel samples, crosses indicate the locations of control points, and the gray area indicates the coding block. (a) Our method. (b) Servaris' [14]. (c) Mathew's [16]. (d) Lakshman's [17].

Receiving the affine-motion matrix $\mathbf{V}$ at the decoder side, the motion field over $\mathcal{X}$ can be derived by (4).

Note that we need to evaluate this motion field at discrete pixel locations for MCP. Attention should be paid to the relation between the block size in pixels and the square size $S \times S$ introduced above. Let $m \times m$ be the size of block in pixels, so the actual pixel locations in the first row, say, are 0 to $m-1$. If we choose the square size to just cover this block, we get $S = (m-1)$ with the next motion square starting at position $(0, m)$ and no overlap between neighboring motion squares. This would create a difficulty in differential motion-vector coding.

Alternatively, we chose to slightly overlap the motion squares with control-point locations as shown in Fig. 3(a), where a coding block is delineated by the gray area. The small circles indicate pixel samples and those with crosses inside indicate locations of control points. The bottom-right one is a nonfree control point and its motion is determined by the affine-motion model derived by $\vec{v}_i, i = 0, 1, 2$. Then, the four control points form a one-cell mesh grid covering $\mathcal{X}$, represented by dashed lines in Fig. 3(a). The mesh grid covers one additional column and one additional row to the right and bottom of the coding block. So, S is then equal to m. Note that the motion at the additional row and column, which belongs to other coding blocks, is not derived from the current mesh grid. There are two advantages of our control-point setting. One is that m is usually some power of 2 which can reduce computation by replacing division in (4) with right shifting. The other and main advantage is that the mesh grids of neighboring blocks are overlapped with each other, which makes the prediction of affine-motion matrices straightforward, as will be discussed in Section II-C.

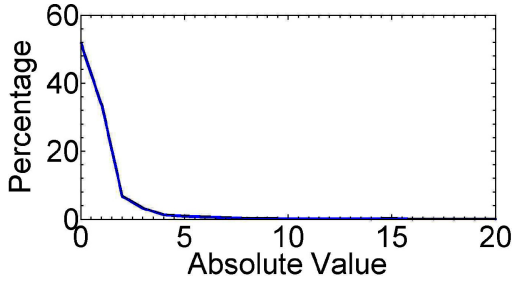


Fig. 4. Statistics of X- and Y-component of $\Delta\vec{v}_{\text{hor}}$ and $\Delta\vec{v}_{\text{ver}}$, obtained by coding *BQSquare*, *BlowingBubble*, *Racehorses*, and *BasketballPass* under the low-delay high-efficiency test condition.

B. Some Notes on Affine-Motion Field

By (4), we know that the affine-motion field varies linearly in (x, y). After discretizing, the MV difference is $\Delta\vec{v}_{\text{hor}} = ((v_{x_1} - v_{x_0})/S, (v_{y_1} - v_{y_0})/S)$ between two consecutive horizontal locations and is $\Delta\vec{v}_{\text{ver}} = ((v_{x_2} - v_{x_0})/S, (v_{y_2} - v_{y_0})/S)$ for vertical ones. To prevent the block being overdeformed and maintain the continuity of the motion field, $\Delta\vec{v}_{\text{hor}}$ and $\Delta\vec{v}_{\text{ver}}$ should not be large. Initially, they were constrained to be less than 1/4 pixel in both directions. This is because the translational MV in our implementation is 1/4 pixel accuracy to which the MVs at control points are also quantized. If $\Delta\vec{v}_{\text{hor}}$ or $\Delta\vec{v}_{\text{ver}}$ is larger than 1/4 pixel, then it can be split into two mesh grids. However, our experiments showed that $\Delta\vec{v}_{\text{hor}}$ and $\Delta\vec{v}_{\text{ver}}$ are usually smaller. An example is shown in Fig. 4. Thus, both horizontal and vertical components of $|\vec{v}_1 - \vec{v}_0|$ and $|\vec{v}_2 - \vec{v}_0|$ are constrained to be less than $S/8$ in this paper.

A scan-line algorithm could be adopted to evaluate the motion field for MCP. Given our control-point setting as described in Section II-A, the MV at the top-left pixel location $(0, 0)$ is equal to $\vec{v}_0$. When moving horizontally, the MV at the current pixel location is obtained by adding $\Delta\vec{v}_{\text{hor}}$ to its left neighbor. For the next line, the MV is obtained by adding $\Delta\vec{v}_{\text{ver}}$ to its neighbor above. By doing so, the computational overload is reduced.

C. Affine-Motion-Matrix Predictors for Differential Coding

Differential coding is usually employed for efficient motion-vector coding. It has been shown in [32] that a competing framework with multiple candidate predictors can significantly improve compression of the motion information. Here, we apply the same concept to code the affine-motion matrix. Our derivation of its predictors is described as follows.

As shown in Fig. 5, motion at A, B, C can be used to predict $\vec{v}_0$, motion at D, E can be used to predict $\vec{v}_1$, and motion at F, G can be used to predict $\vec{v}_2$. More specifically, the mesh grid of the current coding block overlaps those of the neighboring blocks through our chosen control-point representation. The spatial location of $\vec{v}_0$ actually coincides with that of A, B, and C in the neighboring mesh grids. Similarly for $\vec{v}_1$ and $\vec{v}_2$. Denoting P_i as the candidate set for $\vec{v}_i$, we set $P_0 = U\{\vec{v}_A, \vec{v}_B, \vec{v}_C\}$, $P_1 = U\{\vec{v}_D, \vec{v}_E\}$, and $P_2 = U\{\vec{v}_F, \vec{v}_G\}$, where U is defined as an operator that removes replicas. To reduce the overhead of sending the predictor indices, combinations of P_i are obtained by $\Omega = P_0 \times P_1 \times P_2$. Then, only one index is signaled to the decoder instead of 3.

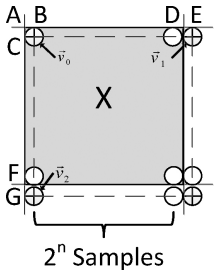


Fig. 5. Predictions for translational MVs at control points.

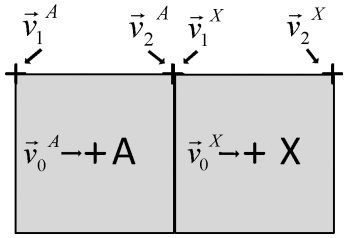


Fig. 6. Example of prediction of affine parameters in [17].

Let $\hat{\mathbf{V}} = (\vec{v}_0, \vec{v}_1, \vec{v}_2) \in \Omega$ be an affine-motion-matrix predictor. Then, $\hat{\mathbf{V}}$ is considered as invalid if any component of $|\vec{v}_1 - \vec{v}_0|$ or $|\vec{v}_2 - \vec{v}_0|$ is greater than $S/8$. The predictor index is unary coded, so the candidate set should be ordered by descendant likelihood for efficient coding. In this paper, the candidate set is arranged in the ascending order of $D(\hat{\mathbf{V}}) = ||\vec{v}_0 - \vec{v}_1||_1 + ||\vec{v}_2 - \vec{v}_1||_1$, a measure of the deformation, since a candidate predictor matrix $\hat{\mathbf{V}}$ is more likely to have a smaller value of $D(\hat{\mathbf{V}})$ [30].

D. Discussion of the Proposed Scheme and Other Methods in the Literature

In [14], affine motion is represented by three MVs at vertices of an equilateral triangle positioned around the centroid of the block as shown in Fig. 3(b). The authors focused on the content-based variable block partition for motion compensation. The control-point representation is proposed for affine-motion estimation and the coding of affine parameters is not discussed. Mathew and Taubman propose an affine-motion model represented through MVs at nominal locations [16] depicted in Fig. 3(c). They focused on the leaf merging technique and its use in wavelet-based scalable video coding. In [17], the affine-motion parameters are denoted by $\vec{v}_0$ at the centroid of the block to represent the translational component, and then two components $d_{\text{left}} = \vec{v}_1 - \vec{v}_0$ and $d_{\text{right}} = \vec{v}_2 - \vec{v}_0$ that represent the relative warping with respect to the centroid. Here, $\vec{v}_1$ and $\vec{v}_2$ are the displacements at the top-left and top-right corners of the current block. For coding, $\vec{v}_0$ is differentially coded as a translational MV using the median predictor as in H.264/AVC [33]. Then, a subset of the neighboring blocks is chosen based on their differences in $\vec{v}_0$ as compared to the current block. The warping parameters of these blocks are mapped to the current block size; then, similarly the medians are used for predicting current warping parameters. As described previously, the prediction of affine parameters in [17] is done separately for the translational component and warping parameters.

Generally, the affine parameter representations could be transformed to each other. The coding methods are the essential differences between our scheme and the others. Assume that the current block $\mathcal{X}$ and its neighboring blocks are in the same moving object since otherwise the prediction from neighbors usually is not accurate. In the case of one affine-motion model in the region, the translational components of $\mathcal{X}$ and its neighboring blocks should have different values due to the deformation of the moving object. In other words, the prediction of the translational component, $\vec{v}_0$ in [17], from the translational component in a neighboring block often will not be accurate. The same problem exists in a direct prediction of the parameters in (2), where the translational component (e, f) is actually the MV at the top-left corner. In the case of several affine-motion models, $\vec{v}_0$ may be well predicted; however, d_{left} and $d_{\text{right)}}$ cannot be accurately predicted as well. Let us say that $\vec{v}_0$ of $\mathcal{X}$ is equal to $\vec{v}_0$ of its left neighbor A for example, as shown in Fig. 6. Note that the motion across the boundary should be continuous, then $\vec{v}_2^A = \vec{v}_1^X$, i.e., $\vec{v}_0^A + d_{\text{right}}^A = \vec{v}_0^X + d_{\text{left}}^X$. It requires that $d_{\text{left}}^A = d_{\text{right}}^A$ for $\vec{v}_0^X = \vec{v}_0^A$ and $d_{\text{left}}^X = d_{\text{left}}^A$. But generally d_{left}^A and d_{right}^A are not equal. Thus, either the motion across the boundary is not smooth or the predictions of $\vec{v}_0$ and d_{left} from A cannot be accurate. Similarly, the prediction of $\vec{v}_0$ and $d_{\text{right)}}$ usually cannot be accurate. Another explanation is that the prediction of d_{left}^X from d_{left}^A is actually predicting $\vec{v}_1^X$ by $\vec{v}_1^A$ (when $\vec{v}_0^X = \vec{v}_0^A$). Now, this prediction may not be accurate since the locations of the two MVs are distanced by the size of A. We would rather use $\vec{v}_2^A$ to predict $\vec{v}_1^X$.

In this paper, differential coding of affine parameters is done by predicting the translational MVs at designated control points. Such a prediction should be accurate as long as the current block and its neighbors belong to the same moving object. Another advantage of the proposed scheme is that the prediction can still work even if warping parameters of all the neighboring blocks are zero due to rate-distortion tradeoff. However, in [17] and other schemes that directly predict the warping parameters, the predictors will have zero values in this case.

III. Implementations and Adaptive Motion Modeling

Our proposed method is implemented within the HEVC test model HM1.0. Analogous to the translational interprediction modes in this test model (coder), four new affine modes are designated, namely *AF_SKIP*, *AF_DIRECT*, *AF_INTER*, and *AF_MERGE*. In this section, we will first make an overview of prediction modes already in HM1.0 and then introduce the new affine modes that are integrated into our version of this coder.

HM1.0 is the first HEVC test model that is established by the joint collaborative team on video coding [34]. It defines a quadtree structure based on a coding unit (CU), an example of which is shown in Fig. 7(a). A CU is a basic unit for compression, a $2N \times 2N$ square block. It may take sizes from 8×8 up to a predefined maximum. Each CU can contain one

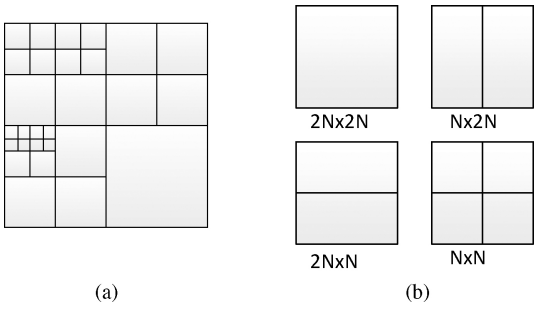


Fig. 7. (a) Quadtree structure of CU. (b) PU structures.

or multiple prediction units (PUs). A PU is a basic unit for intra/interframe prediction and can be one of the four types of partitions: $2N \times 2N$, $2N \times N$, $N \times 2N$, and $N \times N$ as shown in Fig. 7(b). Depending on the PU structure, four interprediction modes and two intraprediction modes (partition $2N \times 2N$ and partition $N \times N$) are defined together with the powerful SKIP and DIRECT modes. Mode decision is performed by operational rate-distortion optimization. Concretely, the coder selects the best mode for a given CU by minimizing

$$J_{\text{mode}} = \text{SSE} + \lambda_{\text{mode}} B_{\text{mode}} \tag{5}$$

where SSE is the sum square error of the reconstructed signal and B_{mode} is the cost of this decision expressed in bits. The Lagrangian parameter λ_{mode} is a constant value determined by a supplied quantization parameter that encodes the chosen quantizer base step size. For variable size adaptation, a full quadtree is first generated; then, a bottom-up pruning algorithm [35] is performed.

The four new affine modes are defined at the CU level and consist of one $2N \times 2N$-type PU. These new affine modes are embedded into the existing rate-distortion optimized mode decision process. Therefore, the encoder adaptively chooses whether to use an affine mode or not based upon its relative mode cost. An additional flag is signaled to the decoder to indicate this mode decision. Specifically, the encoder first encodes the mode information to indicate SKIP, DIRECT, or INTER mode as in the original HM1.0 coder. Then, an additional flag is sent to indicate whether an affine-motion model is used. When an INTER mode and affine-motion model is chosen, another flag is sent to indicate AF_MERGE or AF_INTER mode. In the case of AF_MERGE mode, the merge direction is further signaled. In the following sections, the new affine modes and the interpolation method for affine-motion compensation are described in some detail.

A. Affine SKIP/DIRECT Modes

By exploiting the spatiotemporal correlation between adjacent coding blocks, SKIP and DIRECT modes are powerful tools for improving coding efficiency [36]. Similar to existing translational SKIP and DIRECT modes in HM1.0, two new affine modes, namely AF_SKIP and AF_DIRECT modes are introduced here. Instead of translational MVs, a set of affine-motion-matrix predictor candidates is derived by referring to the motion of nearest-neighbor previously coded partitions as described in Section II-C. Then, a best predictor is chosen by checking all possible candidates and selecting the one that minimizes J_{mode}. This prediction is then used to derive the affine-motion field for the current block. The differences between SKIP and DIRECT modes are as follows [34].

1) The residual signal is coded in the DIRECT mode, but in the SKIP mode, the residual signal is skipped and the prediction signal is used to reconstruct the current block.
2) The interprediction direction is signaled to the decoder in the DIRECT mode, but the bidirectional prediction is always assumed in the SKIP mode.

To be consistent with the HM1.0 test model, the maximum number of affine-motion-matrix predictors is set to five. The efficiency of these affine SKIP and DIRECT modes has been shown in our previous work [30].

B. Affine-Merge Mode

In the new affine-merge mode (AF_MERGE), the mesh cell connection idea proposed [29] is incorporated into the block-based affine-motion model. When merged to the left, $\vec{v}_0$ and $\vec{v}_2$ are derived from the left neighbor, while $\vec{v}_1$ is searched to minimize $J_{\text{motion}} = \text{SAD} + \lambda_{\text{motion}} R(\Delta\vec{v}_1)$. When merged to above, $\vec{v}_0$ and $\vec{v}_1$ are derived from the above neighbor, while $\vec{v}_2$ is searched to minimize $J_{\text{motion}} = \text{SAD} + \lambda_{\text{motion}} R(\Delta\vec{v}_2)$, where SAD is the sum absolute difference of the MCP signal and the rate $R(\Delta\vec{v}_i)$ is the estimated number of bits to code $\Delta\vec{v}_i$. The predictor for $\vec{v}_1$ or $\vec{v}_2$ is chosen from $\vec{v}_0$ and the motion derived by the neighbor grids, i.e., $\{\vec{v}_D, \vec{v}_E\}$ and $\{\vec{v}_F, \vec{v}_G\}$. Coding for $\vec{v}_1$ or $\vec{v}_2$ is the same as that of translational motion. If there is more than one predictor, the predictor index is signaled. Then, the motion difference is arithmetic-coded. The best merge direction is the one that has a smaller value of J_{mode}. Though the word "merging" is used here, it is different from the merging techniques in [12] and [16], where motion in the current block is assumed to be exactly the same as that in the merging target. Here, two MVs of an affine-merge mode block are derived from the neighbor and the third one is differentially coded and transmitted.

C. Affine-Inter Mode

In the new affine-inter mode (AF_INTER), a set of affine-motion-matrix predictor candidates is derived as described in Section II-C. Then, a best predictor is chosen by checking all possible candidates and selecting the one that minimizes $J_p = \text{SAD} + \lambda_{\text{motion}} B_p$. Here, B_p is the number of bits to code the predictor index and $\lambda_{\text{motion}} = \sqrt{\lambda_{\text{mode}}}$. Given the best predictor $\hat{\mathbf{V}}$, the affine-motion matrix $\mathbf{V}^*$ for the current CU is searched by minimizing $J_{\text{motion}} = \text{SAD} + \lambda_{\text{motion}} B_{\text{motion}}$, where B_{motion} is the estimated number of bits to code $\Delta\mathbf{V} = \mathbf{V}^* - \hat{\mathbf{V}}$ and is denoted as $(\Delta\vec{v}_0, \Delta\vec{v}_1, \Delta\vec{v}_2)$. In practice, motion estimation is conducted by the following iteration. If $\hat{\mathbf{V}}$ is not available, then block matching is used to find the initial values.

1) Given $\hat{\mathbf{V}}$, a search region (quadrilateral) is defined. The deformation of the reference block by $\mathbf{V}^*$ is constrained such that it will not exceed the region. Set iteration number $k = 0$.

TABLE I
ENCODER SETTINGS

	LL	RL	LH	RH
Maximum CU size	64			
CU depth	4			
Residual quadtree (RQT) size (min./max.)	4/32			
Maximum RQT depth INTER	2		3	
Maximum RQT depth INTRA	1		3	
Number of reference frames	1 per list			
Luma interpolation	DIF		DCT-IF	
Chroma interpolation	Bilinear interpolation			
Source bit depth	8			
Increased bit depth	0		4	
Entropy coding	CAVLC		CABAC	
Generalized P-slice to B-slice	OFF			
Merge mode	OFF			
Adaptive loop filter	OFF		ON	
Motion search range	64			
Fast search	ON			
Fast encoder decision	ON			
Rate-distortion optimized quantization	ON			
Period of I-frame	only first	32	only first	32
GOP Size	1	8	1	8
Hierarchical B coding	OFF	ON	OFF	ON
Low-delay coding structure	ON	OFF	ON	OFF

2) For $i = 0, 1, 2$, search $\Delta\vec{v}_i$ that minimizes $J_{motion} = \text{SAD} + \lambda_{\text{motion}} R(\Delta\vec{v}_i)$ while keeping the other two fixed at the values obtained in the previous iteration. In this paper, the full search algorithm is adopted to find $\Delta\vec{v}_i$ in a predefined window of range r.
3) $k = k + 1$. Check if either J_{motion} is unchanged or k is greater than a predefined maximum m. If not, go to step 2.

Note that J_{motion} is nonincreasing in each iteration. After the best affine-motion matrix $\mathbf{V}^*$ is found, its predictor is refined by choosing the one that minimizes B_p. To be also consistent with the HM1.0 test model, the maximum number of affine-motion-matrix predictors is set to five and the predictor index is unary coded. If the number of valid predictors exceeds five, we simply discard those with higher $D(\hat{\mathbf{V}})$ values. The residual $\Delta\mathbf{V}$ is coded by sequentially encoding $\Delta\vec{v}_i, i = 0, 1, 2$ using the same motion-vector difference-coding method as in HM1.0.

D. Interpolation Method

Motion in an affine mode block is generally noninteger, which requires interpolation in the reference frame. For the luminance component, a rectangle circumscribes the deformed block in the reference frame that is preinterpolated to 1/4 pixel accuracy using the interpolation filter in HM1.0. Then, the bilinear interpolation is performed within the rectangle if the motion is beyond 1/4 pixel accuracy. For chroma components, the simple bilinear interpolation is adopted as in HM1.0.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we evaluate the proposed block-based affine-motion model with the HM1.0 coder. As described in Section III, four affine modes were designed and incorporated into the HM1.0 coder. We will refer to the new coder as *HM1.0+Affine* and compare it with the original HM1.0 coder. Experiments were conducted with 16 test sequences under low-delay low complexity (LL), random-access low complexity (RL), low-delay high-efficiency (LH), and random-access high-efficiency (RH) test conditions [37]. Some major encoder settings are listed in Table I. The search range for AF_MERGE and AF_INTER modes was set to 15 and 7 units of quarter pixel, respectively. In high-efficiency encoder settings, we used a single context for CABAC coding of each additional flag and a uniform distribution was used for initialization.

TABLE II
BITRATE SAVINGS IN TERMS OF Y BD-RATE

Sequence Name	Resolution	Frames	Y BD-Rate Savings (%)			
			LL	RL	LH	RH
Kimono	1920×1080	240 @ 24 frames/s	0.08	0.02	0.97	1.04
RaceHorses	832×480	300 @ 30 frames/s	3.56	1.01	2.27	0.93
BQMall		600 @ 60 frames/s	5.39	2.18	3.95	1.41
PartyScene		500 @ 50 frames/s	16.59	12.20	5.16	2.28
BasketballDrill		500 @ 50 frames/s	1.45	0.87	2.51	0.64
RaceHorses	416×240	300 @ 30 frames/s	4.33	1.25	2.42	1.00
BQSquare		600 @ 60 frames/s	**33.82**	**23.90**	11.74	4.43
BlowingBubbles		500 @ 50 frames/s	10.35	6.67	4.28	1.93
BasketballPass		500 @ 50 frames/s	1.99	0.28	1.69	0.39
Flowervase		300 @ 30 frames/s	15.03	7.38	**14.26**	4.75
Foreman	352×288	300 @ 30 frames/s	6.40	2.69	4.05	1.91
Mobile		300 @ 30 frames/s	17.00	8.55	6.25	2.76
Flower		250 @ 30 frames/s	11.39	7.38	6.05	**4.89**
Football		260 @ 30 frames/s	1.14	-0.07	1.61	0.46
News		300 @ 30 frames/s	2.14	0.97	4.46	0.82
Stefan		90 @ 30 frames/s	10.14	4.78	5.02	3.53

TABLE III
AVERAGE DECREASES IN THE NUMBER OF CUS AND AVERAGE PERCENTAGE OF SAMPLES CODED BY THE AFFINE MODES

Sequence Name	Decreases in the Number of CUs (%)				Average Affine Mode Coverage (%)			
	LL	RL	LH	RH	LL	RL	LH	RH
Kimono	8.23	6.60	14.71	7.85	17.14	19.64	30.60	17.49
RaceHorses	5.83	3.84	10.23	3.52	21.45	16.22	29.13	11.70
BQMall	11.46	6.42	16.51	5.61	25.26	18.22	34.53	12.73
PartyScene	18.80	14.04	22.16	8.04	48.32	42.89	52.79	17.68
BasketballDrill	6.00	4.42	8.85	3.41	15.88	14.25	26.15	8.64
RaceHorses	7.93	4.08	9.05	3.22	25.72	17.58	27.73	11.11
BQSquare	**23.68**	**22.75**	29.87	12.04	**56.16**	**58.33**	**60.54**	22.06
BlowingBubbles	14.84	9.25	21.01	6.55	39.43	29.43	47.15	17.88
BasketballPass	5.01	3.27	6.79	2.64	18.04	13.55	23.00	8.69
Flowervase	21.94	15.71	**31.20**	12.80	28.63	24.49	43.78	23.87
Foreman	12.05	8.08	19.75	9.70	29.59	26.29	41.60	23.85
Mobile	14.96	10.89	21.91	11.40	49.48	38.44	53.34	24.30
Flower	11.35	12.33	22.70	**15.94**	40.65	35.86	47.84	**32.11**
Football	4.38	2.09	7.68	2.09	17.09	11.23	27.26	10.56
News	8.34	4.43	12.18	4.53	10.94	8.37	15.48	7.17
Stefan	10.54	6.58	20.69	13.36	37.58	30.01	47.84	26.66

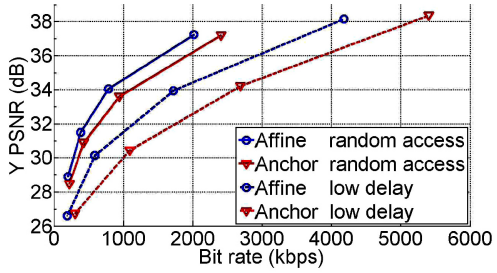
A. Overall Performance

For each test sequence, four rate points were obtained by coding with the QP values {22, 27, 32, 37}, and bitrate savings were measured in terms of Y BD-rate savings [40] as shown in Table II. Depending on the motion types in different sequences, the performance of the new coder varies. Maximum bitrate savings are 33.82% in LL, 23.90% in RL, 14.26% in LH, and 4.89% in RH test conditions. Compared

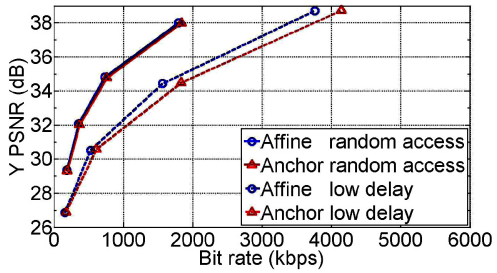
(a)

(b)

Fig. 8. Optical flow obtained by Sun *et al*. [38], [39]. The colors indicate the flow magnitudes and red indicates large motion. (a) Frames 005–006 of *BQSquare*. (b) Frames 001–002 of *Flower*.

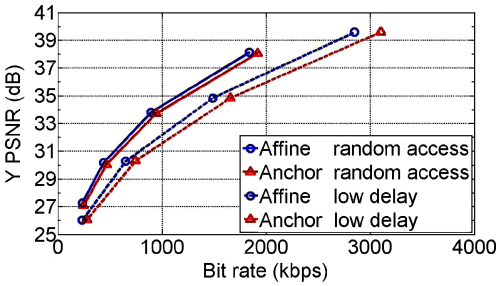


(a)

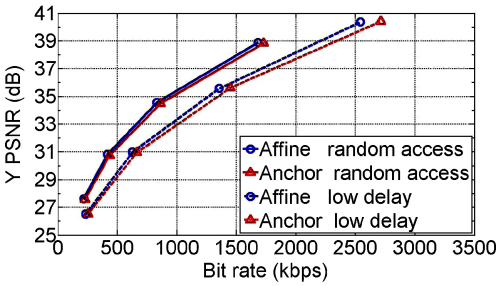


(b)

Fig. 9. RD curves of coding sequence *BQSquare*. The average bitrate savings are 33.82% and 23.90% for low-delay and random-access test conditions in low-complexity encoder settings, and 11.74% and 4.43% in high-efficiency encoder settings. (a) Low complexity. (b) High efficiency.



(a)



(b)

Fig. 10. RD curves of coding sequence *Flower*. The average bitrate savings are 11.39% and 7.38% for low-delay and random-access test conditions in low-complexity encoder settings, and 6.05% and 4.89% in high-efficiency encoder settings. (a) Low complexity. (b) High efficiency.

to the original HM1.0 coder, the number of CUs produced by the HM1.0+Affine coder is decreased for all test sequences and test conditions as shown in Table III. This demonstrates that larger blocks are chosen more often by enabling affine-motion compensation, which also explains the gains obtained by the new coder. Generally, bitrate reduction is high when there is the larger decrease in the number of CUs and vice versa. The usage of the affine modes or coverage, i.e., the proportion of the time they are chosen, averaging over all frames, and for each sequence, is shown in the right portion of Table III. Again, we can observe that generally, the larger the percentage of samples coded by affine modes, the greater the bitrate reduction achieved.

In sequences *BQSquare* and *Flower*, the motion is mostly caused by camera panning and changes linearly according to the depth of the objects. Two examples are shown in Fig. 8. Such motion can be well captured by the affine-motion model. Thus, bitrate reductions achieved by HM1.0+Affine are significant. The peak signal-to-noise ratio performance curves are plotted in Figs. 9 and 10. As can be seen from the figures, the new coder outperforms the original HM1.0 coder at all rate points under all test conditions. In the sequence *BQSquare*, the affine mode coverage amounts to 60% of the whole clip. An example is shown in Fig. 11, where the magenta areas indicate coding in affine modes. An interesting phenomenon is that the affine modes tend to appear on the object boundaries (red sun umbrellas) and textured regions. On the other hand, the smooth areas, where the translational motion compensation is sufficient, are not coded by an affine mode.

In the head-and-shoulder sequence *News*, however, much of the scene is static background and the motion of the ballet dancers displayed on the background TV screen is hard to capture. Therefore, the bitrate saving for this sequence is small. In *Football*, the motion caused by the fast movement of players and camera cannot be well estimated. The use of affine-motion model also does not help much. A tiny loss is found under the RL test condition due to the overhead of sending additional flags. In such sequences, one can simply turn OFF the affine modes. Note that there is no performance loss under

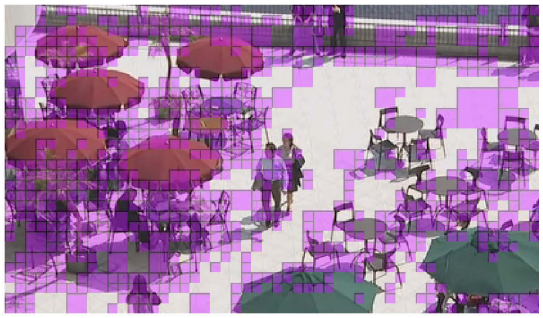


Fig. 11. Affine mode coverage in the coding frame 205 of *BQSquare* with $QP = 22$ at the low-delay low-complexity condition.

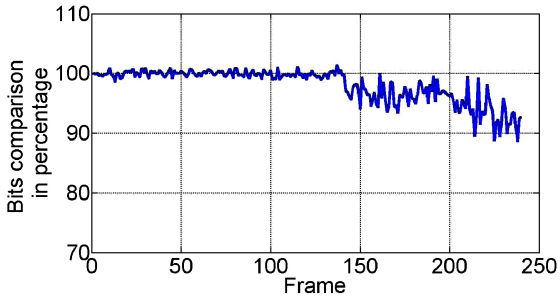


Fig. 12. Bit comparison between HM1.0+Affine and *HM1.0* in coding *Kimono* with $QP = 22$ at the low-delay low-complexity condition.

high-efficiency encoder settings and this is due to the more efficient coding of overhead by CABAC.

As shown in Tables II and III, the performance of HM1.0+Affine depends on the motion types in different sequences. It is also true within a sequence. In Fig. 12, we show a comparison of the relative number of bits to code each frame of the *Kimono* sequence between the two coders. Specifically, we plot the ratio of the number of bits output by the HM1.0+Affine coder divided by the number of bits output by the HM1.0 test model coder, and then multiply by 100 to get a percent. In the first 140 frames, the camera is moving parallel to the lady and the background trees are shifting to the left. Therefore, the motion is mostly translational and the HM1.0+Affine coder uses almost the same number of bits to code these frames, i.e., the curve is around 100%. In the last 100 frames, the camera is moving closer to the house in the background. Thus, deformation appears and the curve generally drops, indicating that these deformations are captured by our affine-motion model. Thus, further improvements could be achieved by adaptively switching ON and OFF at the slice level.

In Table IV, our HM1.0+Affine coder is compared with the coder proposed in [21], denoted as *HM1.0+PSKIP*. In *HM1.0+PSKIP*, a parametric SKIP mode based on global-motion estimation is designed and embedded into the HM1.0 coder. We used the low-complexity encoder settings as in [21] to code five sequences. A comparison of Y BD-rate savings achieved by the two coders is shown in Table IV, showing that our HM1.0+Affine coder outperforms the *HM+PSKIP* coder in most cases.

TABLE IV
COMPARISON OF Y BD-RATE SAVINGS ACHIEVED BY HM1.0+PSKIP AND HM1.0+AFFINE

Sequence	HM1.0+PSKIP (%)		HM1.0+Affine (%)	
	LL	RL	LL	RL
BQSquare	2.8	3.6	29.04	22.68
City	3.6	0.5	4.73	1.88
Entertainment	0.0	0.0	2.78	1.80
PartyScene	2.3	2.8	12.92	11.38
Station2	29.1	9.7	10.73	12.52

TABLE V
CONTRIBUTION AND COMPLEXITY OF AFFINE MODE

	Y BD-Rate Savings (%)				Encoding Time (%)			
	LL	RL	LH	RH	LL	RL	LH	RH
Coder1	16.55	8.95	6.05	1.93	101.01	100.57	102.07	99.78
Coder2	25.62	15.80	11.05	3.61	206.12	207.69	199.14	202.99
Coder3	27.36	18.99	12.40	4.01	307.24	291.95	291.44	282.50
Coder2*	25.51	15.68	10.96	3.51	129.85	137.66	129.52	135.64
Coder3*	27.35	19.07	12.50	4.08	154.44	167.64	151.80	164.51
Coder4*	23.12	15.15	10.51	2.75	122.94	130.24	121.88	126.11

B. Contribution and Complexity of Affine Mode

Another experiment was set up to illustrate the contribution of each affine mode and its complexity. In this experiment, the first 100 frames of *BQSquare* were used for testing. For encoder settings, fast search and fast encoder decision were turned OFF. The experiment was conducted on an isolated Windows 7 PC with Intel Core 2 3.0-GHz CPU and 3-GB RAM. The results are shown in Table V, with the anchors produced by the original HM1.0 coder.

In *Coder1*, the AF_SKIP and AF_DIRECT modes are added to the HM1.0 coder. It is shown that these two modes alone provide significant bitrate reductions for this test clip without much computational overload. In *Coder2*, the coding performance is further improved by adding the AF_MERGE mode. However, the encoder time is nearly doubled. When the AF_INTER mode is finally added in *Coder3*, thus the HM1.0+Affine coder, only 0.4%−3.0% more bitrate reduction are achieved. But the encoder time is further increased by about 80%−100%. Comparing the results of *Coder3* in Table V to that in Table II, it is noticed that the differences are only 6.46% and 4.91% points[1] for the low-complexity test conditions and 0.66 and 0.42 for the high-efficiency conditions. Thus, the fast search and fast encoder decision have minor influence on the performance of the proposed scheme.

We experimentally found out that $\Delta\vec{v}_i$ is usually small in both AF_MERGE and AF_INTER modes, so the search ranges could be reduced. The superscript $*$ in Table V indicates that the search range is reduced to seven and three units of quarter pixel for these two modes, respectively. In *Coder2** and *Coder3**, the encoding time for this test clip is significantly reduced compared to *Coder2* and *Coder3*, respectively, but the performance loss is negligible. Though the AF_INTER mode does not contribute much to the performance of HM1.0+Affine coder, in *Coder4**, we show that

[1]Relatively small compared to 33.82% and 23.90%.

the performance of AF_INTER mode itself is similar to that of Coder2* and Coder3*.

Depending on the usage of affine-motion compensation, the decoding time of HM1.0+Affine can be up to three times of that in original HM1.0. The computational overload comes from the two passes in our interpolation method. That is, a reference region is first preinterpolated into 1/4 pixel accuracy, and then followed by the bilinear interpolation. The reference region is the minimum rectangle that circumscribes the deformed block. In this research coder, we have implemented the affine model of (4) at full accuracy. It is expected that more efficiency savings can result from reduced accuracy in this area.

V. Conclusion

An affine-motion model can provide better MCP than a conventional translational motion model. However, two problems need to be solved when this is applied to the video coding application. One is the estimation of affine parameters, considering that not only accuracy but speed is required. Another problem is the efficient coding of the affine parameters which is addressed in this paper. We proposed a new control-point representation for the affine-motion model which makes it easy to differentially encode the affine-motion matrix. Such a scheme was proved efficient by placing four new affine modes into the HEVC test model HM1.0 coder. The proposed affine-motion model is general and can also be applied to other block-based coders as well.

The estimation of the affine-motion parameters was not specifically discussed in this paper. An iterative greedy algorithm was adopted to refine the affine-motion matrix. It is suboptimal and time consuming. However, the computational overload could be reduced by constraining the search range. We experimentally found that $\Delta\vec{v}_i$ is usually very small. More advanced estimation methods can be investigated for further improvement. An extension of the fast motion estimation method in block matching, enhanced predictive zonal search [41], for example, may be an option to speed up the estimation. It also may be possible to simultaneously search $\Delta\vec{v}_i$ instead of our current iterative greedy algorithm.

Acknowledgment

The authors would like to thank the anonymous reviewers for their efficient review and helpful comments.

References

[1] G. J. Sullivan and T. Wiegand, "Video compression—From concepts to the H.264/AVC standard," *Proc. IEEE*, vol. 93, no. 1, pp. 18–31, Jan. 2005.

[2] J. W. Woods, *Multidimensional Signal, Image, and Video Processing and Coding*. New York: Academic, 2011.

[3] Y. Nakaya and H. Harashima, "An iterative motion estimation method using triangular patches for motion compensation," in *Proc. Vis. Commun. Image Process.*, vol. 1605. 1991, pp. 546–557.

[4] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 3, pp. 339–356, Jun. 1994.

[5] G. J. Sullivan and R. L. Baker, "Motion compensation for video compression using control grid interpolation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 1991, pp. 2713–2716.

[6] C.-L. Huang and C.-Y. Hsu, "A new motion compensation method for image sequence coding using hierarchical grid interpolation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 1, pp. 42–52, Feb. 1994.

[7] M. Yazdi and A. Zaccarin, "Interframe coding using deformable triangles of variable size," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 1997, pp. 456–459.

[8] Y. Altunbasak and A. M. Tekalp, "Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes," *IEEE Trans. Image Process.*, vol. 6, no. 9, pp. 1255–1269, Sep. 1997.

[9] Y. Altunbasak and A. M. Tekalp, "Occlusion-adaptive, content-based mesh design and forward tracking," *IEEE Trans. Image Process.*, vol. 6, no. 9, pp. 1270–1280, Sep. 1997.

[10] G. Al-Regib, Y. Altunbasak, and R. M. Mersereau, "Hierarchical motion estimation with content-based meshes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 10, pp. 1000–1005, Oct. 2003.

[11] V. Seferidis and M. Ghanbari, "Generalised block-matching motion estimation using quad-tree structured spatial decomposition," in *Proc. IEE Vis. Image Signal Process.*, vol. 141, no. 6, pp. 446–452, Dec. 1994.

[12] M. Karczewicz, J. Nieweglowski, and P. Haavisto, "Video coding using motion compensation with polynomial motion vector fields," *Signal Process.: Image Commun.*, vol. 5965, nos. 1–3, pp. 63–91, 1997.

[13] H. Li and R. Forchheimer, "A transformed block-based motion compensation technique," *IEEE Trans. Commun.*, vol. 43, no. 2, pp. 1673–1676, Feb.–Mar.–Apr. 1995.

[14] M. Servais, T. Vlachos, and T. Davies, "Motion Compensation using content-based variable-size block-matching," in *Proc. Picture Coding Symp.*, 2004, pp. 1–6.

[15] R. C. Kordasiewicz, M. D. Gallant, and S. Shirani, "Encoding of affine motion vectors," *IEEE Trans. Multimedia*, vol. 9, no. 7, pp. 1346–1356, Nov. 2007.

[16] 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, Oct. 2010.

[17] H. Lakshman, H. Schwarz, and T. Wiegand, "Adaptive motion model selection using a cubic spline based estimation framework," in *Proc. 17th IEEE Int. Conf. Image Process.*, Sep. 2010, pp. 805–808.

[18] T. Wiegand, E. Steinbach, and B. Girod, "Affine multipicture motion-compensated prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 2, pp. 197–209, Feb. 2005.

[19] X. Li, J. R. Jackson, A. K. Katsaggelos, and R. M. Mersereau, "Multiple global affine motion model for H.264 video coding with low bit rate," in *Proc. SPIE Image Video Commun. Process.*, vol. 5685. Jan. 2005, pp. 185–194.

[20] H. Yu, Z. Lin, and F. C. T. Teo, "An efficient coding scheme based on image alignment for H.264/AVC," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3. May 2009, pp. 629–632.

[21] A. Glantz, M. Tok, A. Krutz, and T. Sikora, "A block-adaptive skip mode for inter prediction based on parametric motion models," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 1201–1204.

[22] R. C. Kordasiewicz, M. D. Gallant, and S. Shirani, "Affine motion prediction based on translational motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1388–1394, Oct. 2007.

[23] H.-K. Cheung and W.-C. Siu, "Local affine motion prediction for H.264 without extra overhead," in *Proc. IEEE Int. Symp. Circuits Syst.*, May–Jun. 2010, pp. 1555–1558.

[24] A. Muhit, M. Pickering, M. Frater, and J. Arnold, "Video coding using elastic motion model and larger blocks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 5, pp. 661–672, May 2010.

[25] H. Yuan, Y. Chang, Z. Lu, and Y. Ma, "Model based motion vector predictor for zoom motion," *IEEE Signal Process. Lett.*, vol. 17, no. 9, pp. 787–790, Sep. 2010.

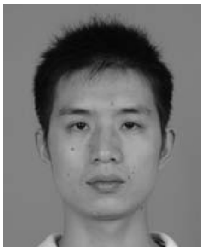
[26] H. Yuan, J. Liu, J. Sun, H. Liu, and Y. Li, "Affine model based motion compensation prediction for zoom," *IEEE Trans. Multimedia*, vol. 14, no. 4, pp. 1370–1375, Aug. 2012.

[27] L.-M. Po, K.-M. Wong, K.-W. Cheung, and K.-H. Ng, "Subsampled block-matching for zoom motion compensated prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, pp. 1625–1637, Nov. 2010.

[28] H.-S. Kim, J.-H. Lee, C.-K. Kim, and B.-G. Kim, "Zoom motion estimation using block-based fast local area scaling," *IEEE Trans. Circuits Syst. Video Technol*, vol. 22, no. 9, pp. 1280–1291, Sep. 2012.

[29] H. Huang, J. W. Woods, and Y. Zhao, "Motion compensated prediction using partial mesh generation," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 1677–1680.

[30] H. Huang, J. Woods, Y. Zhao, and H. Bai, "Affine skip and direct modes for efficient video coding," in *Proc. IEEE Vis. Commun. Image Process.*, Nov. 2012, pp. 1–6.
[31] R. C. Kordasiewicz and M. D. Gallant, "Modeling quantization of affine motion vector coefficients," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 1, pp. 86–97, Jan. 2007.
[32] G. Laroche, J. Jung, and B. Pesquet-Popescu, "RD optimized coding for motion vector predictor selection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 12, pp. 1681–1691, Dec. 2008.
[33] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[34] K. McCann, B. Bross, and S. Sekiguchi, "High efficiency video coding (HEVC) test model 1 (HM 1) encoder description," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Guangzhou, China, Tech. Rep. JCTVC-C402, Oct 2010.
[35] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Trans. Image Process.*, vol. 3, no. 3, pp. 327–331, May 1994.
[36] A. M. Tourapis, F. Wu, and S. Li, "Direct mode coding for bipredictive slices in the H. 264 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 1, pp. 119–126, Jan. 2005.
[37] F. Bossen, "Common test conditions and software reference configurations," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, Guangzhou, China, Tech. Rep. JCTVC-C500, Oct. 2010.
[38] D. Sun, *Optical flow MATLAB code* (2010) [Online]. Available: http://www.cs.brown.edu/people/dqsun/research/software.html
[39] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2432–2439.
[40] G. Bjontegaard, "Improvements of the BD-PSNR model," ITU-T Video Coding Experts Group (VCEG), Heinrich-Hertz-Institute, Berlin, Germany, Tech. Rep. VCEG-AI11, Jul. 2008.
[41] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Proc. SPIE Vis. Commun. Image Process.*, Jan. 2002, pp. 1069–1079.



Han Huang (S'11) received the B.S. degree in computer science from Beijing Jiaotong University, Beijing, China, in 2007, where he is currently pursuing the Ph.D. degree in signal and information processing at the Institute of Information Science.

From 2009 to 2011, he was a visiting student in the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA. His current research interests include video compression and processing.

John W. Woods (M'70–SM'83–F'88) received the B.S., M.S., and Ph.D. degrees in electronic engineering from the Massachusetts Institute of Technology, Cambridge, in 1965, 1968, and 1970, respectively.

Since 1976, he has been with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA, where he is currently a Professor. His current research is concentrated on robust and scalable video coding for networks. He has coauthored a textbook *Probability, Statistics, and Random Processes for Engineers* with H. Stark (4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2012) and one graduate textbook *Multidimensional Signal, Image, and Video Processing and Coding* (1st ed. New York: Academic, 2006) with a second edition in 2012. His current research interests include image and video estimation, restoration, filtering, and specially video compression coding. He has authored or coauthored more than 100 papers in these fields.

Dr. Woods was the corecipient of the 1976 and 1987 Senior Paper Award of what is now the IEEE Signal Processing Society. He served as the Technical Program Co-Chairman for the 1st IEEE International Conference on Image Processing in 1994. He received a Technical Achievement Award from the IEEE Signal Processing Society in 1994. He was a founding member of the Editorial Board of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He received an IEEE Centennial Medal in 2001. From 2001 to 2006, he was a member of the ISO standards Committee on MPEG, and a member of the Digital Cinema Initiatives Compression Committee in 2004.

Yao Zhao (M'06–SM'12) received the B.S. degree from Fuzhou University, Fuzhou, China, in 1989, and the M.E. degree from Southeast University, Nanjing, China, in 1992, both in radio engineering, and the Ph.D. degree from the Institute of Information Science, Beijing Jiaotong University (BJTU), Beijing, China, in 1996.

In 1998, he was an Associate Professor at BJTU, where he became a Professor in 2001. From 2001 to 2002, he was a Senior Research Fellow in the Information and Communication Theory Group, Faculty of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands. He is currently the Director of the Institute of Information Science, BJTU. He is currently leading several national research projects such as 973 Program, 863 Program, and the National Science Foundation of China. His current research interests include image/video coding, digital watermarking and forensics, and video analysis and understanding.

Dr. Zhao serves on the editorial boards of several international journals, including as an Area Editor of Signal Processing: Image Communication (Elsevier), and as an Associate Editor of Circuits, System & Signal Processing (Springer). He was the recipient of National Science Foundation of China for Distinguished Young Scholars in 2010.

Huihui Bai received the Ph.D. degree in signal and information processing from Beijing Jiaotong University (BJTU), Beijing, China, in 2008.

She is currently an Associate Professor at the Institute of Information Science, BJTU. She has been involved in research and development work in video coding technologies and standards such as high-efficiency video coding, 3-D video compression, multiple description video coding, and distributed video coding. She is leading or participating in several research projects such as 973 Program, 863 Program, the National Natural Science Foundation of China, Beijing Natural Science Foundation, and Jiangsu Provincial Natural Science Foundation.