ADAPTIVE INTERPOLATED MOTION COMPENSATED PREDICTION

Wei-Ting Lin, Tejaswi Nanjundaswamy, Kenneth Rose

Department of Electrical and Computer Engineering, University of California Santa Barbara, CA 93106
Email: {weiting, tejaswi, rose}@ece.ucsb.edu

ABSTRACT

Current video coders rely heavily on block-based motion compensation, which is known to accurately capture pure translation, but to (at best) approximate all other types of motion, such as rotation and zoom. Moreover, as motion vectors are obtained through pixel-domain block matching to optimize a rate-distortion cost, and do not necessarily represent the actual motion, the model should not be considered a proper sampling of the underlying pixel motion field. This paper explicitly treats several neighboring motion vectors as pointers to multiple observation sources for estimating a pixel in the current frame. The corresponding optimal linear estimation coefficients are derived for predicting each pixel, given the observations obtained based on nearby motion vectors. Prediction coefficients are further adapted to local statistics by switching between predefined sets of coefficients, which are trained offline through a procedure of "K-modes" clustering. Experimental results outside the training set validate this paradigm with significant bit rate savings over conventional motion compensated prediction.

***Index Terms*—** Video coding, motion compensation, adaptation, linear predictor

1. INTRODUCTION

Motion-compensation is one of the key components in video coding. It is based on the assumption that each pixel value in the current frame is correlated to some pixel in the previously coded frames. Therefore, instead of encoding the raw pixel values, pixels are predicted from reference frames, and only the prediction errors are encoded. The difference in position between the target and reference pixel is referred to as a motion vector, which has to also be coded and sent to the decoder. Since the complexity for searching and signaling overhead for transmitting the motion vector at each pixel would outweigh the benefits of exploiting this temporal redundancy, modern video coding standards, such as HEVC [1] and VP9 [2], use block-based motion compensation (BMC) to exploit temporal redundancies. These coders divide a frame into non-overlapping blocks, which are predicted from similar blocks in the reference frame, to minimize the rate-distortion (RD) cost. BMC implicitly assumes that all pixels in a block move uniformly, i.e., the motion is pure translation. This assumption does not hold in a number of scenarios, e.g., a block covering multiple objects that differ in their motion, or non-translation motion components such as rotation and zoom. Thus, BMC may result in large prediction errors, as well as annoying blocking artifacts.

Variable block size motion compensation was proposed to enhance the prediction accuracy by a more flexible partition of the frame into blocks [3], but this still does not fully account for actual object shapes. Employing higher order models to capture motion other than a simple translation was proposed in [4], wherein motion vectors estimated by BMC are viewed as control points to construct smoothly varying motion vectors across pixels. This allows pixels in the same block to have different motion vectors, thus potentially capturing other types of motion, such as rotation and zoom. However, in the presence of objects moving in different directions, the motions at control points will be a compromised choice, rendering the approach ineffective, and in scenarios where adjacent blocks predict from different reference frames, the approach is simply inapplicable. Other limitations involve the subpixel precision of many motion vectors of individual pixel and the need to rely on a finite set of imperfect interpolation filters, which yield errors, see [5].

The above two approaches still try to associate each block or pixel with a single most suitable motion vector to construct the prediction. Instead of limiting the approach to a single motion vector per pixel, we propose to use the neighboring blocks' motion vectors as pointers to sources of relevant observations, and construct the final prediction using optimal linear estimation coefficients to combine these observations. Relevant early work includes the overlapped block motion compensation (OBMC) approach, which was first proposed to reduce blocky artifacts [6, 7], and was later observed to also reduce the prediction error. In [8], an estimation approach to design the weighting window based on a symmetric motion assumption, was proposed, and in [9] a parametric window design based on the motion model was proposed. These approaches use a single type of extended window such that overlapping neighboring windows effectively average their respective observations. In this work the focus is on implementing the optimal linear estimator for each pixel from ob-

This work was supported by Google Inc.

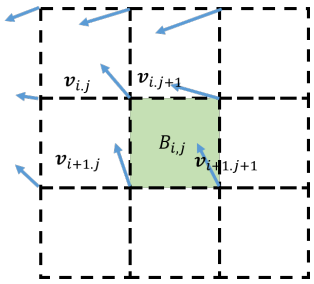


Fig. 1: Grid of motion vectors.

servations obtained by applying the multiple nearby motion vectors available. Moreover, the estimator is adaptive to variation to local statistics by switching between K sets of coefficients that are designed offline based on training data. We re-emphasize that, unlike prior methods that design the weights for a window *centered* around a motion vector position, we design the sets of coefficients for the area that lies *between* motion vector positions. This distinction allows us to accurately capture variations in how predictions due to neighboring motion vectors need to be weighted, and in effect enables accounting for arbitrary object shapes. We design the weights via K-modes clustering to capture the variation in local statistics. Note that additional side information needs to be transmitted to indicate the selected set of coefficients per block, hence, K is chosen to balance the trade off between prediction accuracy and rate overhead. Experimental results demonstrate the efficacy of the proposed paradigm with average 8.46% bit rate savings over conventional fixed block motion compensation.

2. PROPOSED PREDICTION MODEL

Conventional motion compensated prediction for pixel $\mathbf{s}$ in a block at location (i,j) in frame k, can be written as

$$\tilde{x}_k(\mathbf{s}) = \hat{x}_{k-1}(\mathbf{s} - \mathbf{v}_{i,j}), \tag{1}$$

where, $\mathbf{v}_{i,j}$ is the motion vector for the (i,j) block, and $\hat{x}_{k-1}(\cdot)$ is a reconstructed pixel in the previous frame. (We assume without loss of generality that the reference block is in the previous frame). Since a single motion vector cannot capture complex motions within a block, we propose to exploit nearby motion vectors to obtain additional observations, and generate the final prediction by linearly combining the observations weighted by appropriate coefficients. The coefficients are selected from one of the predefined K-sets that are designed to capture variations in the local statistics.

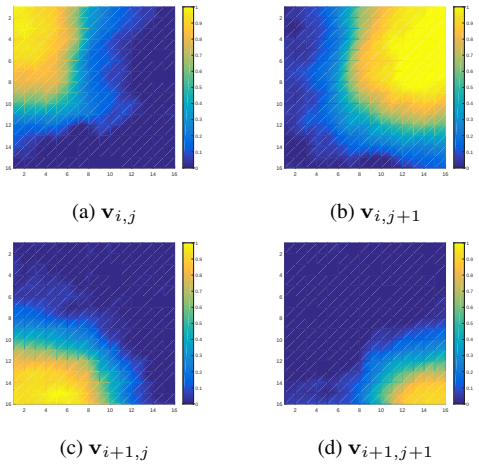
We denote by $B_{i,j}$ the block of pixels lying *between* motion vectors $\mathbf{v}_{i,j}, \mathbf{v}_{i+1,j}, \mathbf{v}_{i,j+1}$ and $\mathbf{v}_{i+1,j+1}$, as shown in Fig. 1. If these motion vectors were produced by conventional BMC then each corresponds to the center of its block. Hence, our block definition is off-grid and covers one quadrant each from four blocks of the standard fixed block grid. Let $\mathbf{s}_{i,j}^{tl}$ be the top-left pixel in $B_{i,j}$, and $\mathbf{s}' = \mathbf{s} - \mathbf{s}_{i,j}^{tl}$, be the relative position within the block. The overall prediction for the pixel $\mathbf{s} \in B_{i,j}$ in frame k is calculated as

$$\tilde{x}_k(\mathbf{s}) = \sum_{m=0}^{1}\sum_{n=0}^{1} c_{m,n}^{q}(\mathbf{s}')\hat{x}_{k-1}(\mathbf{s} - \mathbf{v}_{i+m,j+n}) \tag{2}$$

$$= \mathbf{c}^{q}(\mathbf{s}')^{\intercal}\hat{\mathbf{x}}_{k-1}(\mathbf{s}), \tag{3}$$

where $c_{m,n}^{q}(\mathbf{s}')$ is the q-th set coefficient for prediction at position $\mathbf{s}'$ using the corresponding (m,n) neighboring motion vector. Equation (2) is shown in vector form in (3) where $^{\intercal}$ denotes transposition. The set of coefficients is selected to minimize the mean squared prediction error, i.e.,

$$q = \underset{r\in\{0,\ldots,K-1\}}{\arg\min} \sum_{\mathbf{s}\in B_{i,j}} \Big(x_k(\mathbf{s}) - \mathbf{c}^{r}(\mathbf{s}')^{\intercal}\hat{\mathbf{x}}_{k-1}(\mathbf{s}) \Big)^2. \tag{4}$$



(a) $\mathbf{v}_{i,j}$ (b) $\mathbf{v}_{i,j+1}$ (c) $\mathbf{v}_{i+1,j}$ (d) $\mathbf{v}_{i+1,j+1}$

Fig. 2: An example set of weight distributions corresponding to motion vectors $\mathbf{v}_{i,j}, \mathbf{v}_{i+1,j}, \mathbf{v}_{i,j+1}$ and $\mathbf{v}_{i+1,j+1}$

An example set of coefficients is shown in Fig. 2. The coefficients tend to approach one near the corresponding motion vector position and decrease with distance. As discussed in Sec. 1, applying coefficients this way allows us to capture variations in local statistics corresponding to significance of predictions due to neighboring motion vectors. Ideally, different coefficients that are optimal for each block can be used, but these coefficients must be known to the decoder as well. In order to implement the proposed prediction model without introducing too much signaling overhead, we restrict ourselves to using K sets of coefficients. These coefficients are stored in both the encoder and decoder, hence, we only need to signal the index to the decoder.

3. COEFFICIENT DESIGN VIA K-MODES CLUSTERING ALGORITHM

We propose to design the coefficients offline through a "K-modes" clustering-based approach. Note that the overall ob-

jective of the coder is to optimize the tradeoff between rate and quantization error, and the quantized pixel is the sum of prediction $\tilde{x}_k(\mathbf{s})$ and quantized prediction error $\hat{e}_k(\mathbf{s})$, i.e.,

$$\hat{x}_k(\mathbf{s}) = \tilde{x}_k(\mathbf{s}) + \hat{e}_k(\mathbf{s}). \quad (5)$$

Designing coefficients to minimize prediction error leads to better prediction, but this does not guarantee better reconstruction. Hence we propose to design the coefficients while accounting for the reconstruction error.

Once blocks are classified into K clusters based on (4), the squared reconstruction error for each cluster C_q is

$$J = \sum_{B_{i,j} \in C_q} \sum_{\mathbf{s} \in B_{i,j}} \Big(x_k(\mathbf{s}) - \tilde{x}_k(\mathbf{s}) - \hat{e}_k(\mathbf{s}) \Big)^2. \quad (6)$$

Given the discreet nature of quantization, this cost (6) is piecewise continuous in the prediction coefficients. Sufficiently small changes in coefficient values will (almost always) only affect the reconstructed value through the prediction term of (5), hence optimal predictive coefficients $\mathbf{c}^q(\mathbf{s}')$ must satisfy

$$\mathbf{c}^q(\mathbf{s}') = E[\hat{\mathbf{x}}_{k-1}(\mathbf{s})\hat{\mathbf{x}}_{k-1}(\mathbf{s})^\intercal]^{-1} E[x'_k(\mathbf{s})\hat{\mathbf{x}}_{k-1}(\mathbf{s})], \quad (7)$$

where $x'_k(\mathbf{s}) = x_k(\mathbf{s}) - \hat{e}_k(\mathbf{s})$. Overall an iterative closed-loop approach is used to update these values until convergence:

1. Given the sets of coefficients at iteration $i-1$, a training set of reconstructions $\{\hat{x}_0^{(i)}, \hat{x}_1^{(i)}, \hat{x}_2^{(i)}, \cdots, \hat{x}_N^{(i)}\}$ and quantized prediction errors $\{\hat{e}_0^{(i)}, \hat{e}_1^{(i)}, \hat{e}_2^{(i)} \cdots \hat{e}_N^{(i)}\}$ are generated for iteration i using (4).

2. Given the new training set, (7) is employed to compute the new coefficients.

3.1. Motion refinement for interpolated prediction

In the proposed interpolated prediction framework, each motion vector influences multiple prediction blocks, which implies the motion vectors cannot be optimally selected independently. Hence, we propose an iterative motion refinement algorithm.

Given the coefficients for all the K modes, we initialize the motion vector for each block $B_{i,j}$ via conventional motion compensation, and then update the motion vectors as follows:

1. Calculate the optimal mode for each block $B_{i,j}$ given the motion vectors.

2. Fix the modes and $B_{i,j}$'s neighboring blocks' motion vectors; run motion search to minimize the rate-distortion cost.

The above two steps are repeated until convergence. We note that the motion vector update in Step 2 above can be divided into different groups to be run in parallel, since a motion vector only affects a limited area (at maximum four blocks for fixed block sizes). For example, as shown in Fig. 3, all motion vectors shown with the same color can be updated in parallel.

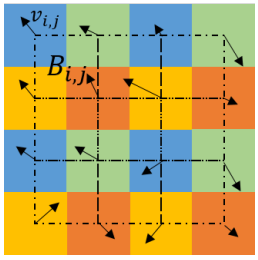


Fig. 3: Motion vectors of blocks shown with the same color can be updated in parallel during motion refinement for interpolated prediction.

4. EXPERIMENTAL RESULTS

We evaluated the proposed approach in the experimental branch of the VP9 framework. It is important to emphasize that the proposed paradigm is applicable to any modern video coding standard, as they all employ variants of BMC. For simplicity of simulations, we restrict the coder to use 16×16 fixed block size in an *IPPP* structure, with only the previous frame allowed as reference for inter prediction. To minimize complexity overhead, we limit the motion refinement to one iteration and the search window size to $[-1, 1]$ on both horizontal and vertical directions. The coefficients are initialized using 2-D raised cosine function, which is defined as

$$H_{2D}(\beta_x, \beta_y, x, y) = C(x, y) H_{1D}(\beta_x, x) H_{1D}(\beta_y, y),$$

where $H_{1D}(\beta_x, x)$ is the 1-D raised cosine function,

$$H_{1D}(\beta, x) =$$

$$\begin{cases} 1, 0 \leq |x| \leq \dfrac{(1-\beta)B}{2} \\ \dfrac{1}{2} + \dfrac{1}{2} \cos\left(\dfrac{\pi B}{\beta} \left[x - \dfrac{(1-\beta)B}{2} \right] \right), \dfrac{(1-\beta)B}{2} < |x| \\ 0, \text{otherwise} \end{cases}$$

and $C(x, y)$ is the normalization function. We selected $\beta_x, \beta_y \in \{0, 0.5, 1\}$ (i.e. $K = 9$) and the initial coefficients are uniformly sampled values of the function $H_{2D}(\beta_x, \beta_y, x, y)$ for $0 \leq x, y \leq 1$. We design separate set of coefficients for different target bit-rate regions and different range of resolutions. The training set for CIF resolution consists of first 100 frames of *Flower, Coastguard, Mobile* and *Stefan* video sequences, and the training set for HD resolution consists of first 20 frames of *BQTerrace, Cactus, In_to_Tree* and *Pedestrian* video sequences. The trained

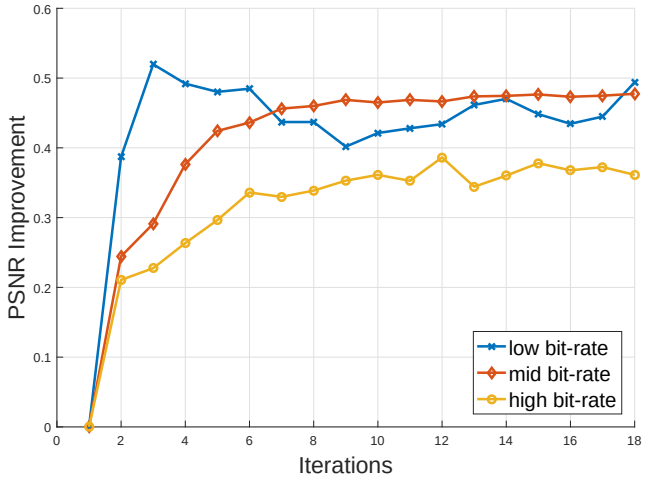


Fig. 4: PSNR improvement (in dB) versus iterations of the proposed K-mode clustering algorithm for different target bit rate regions.

coefficients are stored in both the encoder and decoder. The mode index is entropy coded and the signaling overhead is accounted for in the results.

Fig. 4 shows the average PSNR improvements of the training set at each iteration for different target bit-rate regions. In the low bit-rate case, most of the prediction residuals are quantized to zero, which implies that improvement in prediction accuracy directly maps to improvement of reconstruction, and hence we see significant improvement in the first few iterations. This is not the case in mid and high target bit-rate regions, and therefore the improvement is much more smooth. Note that the low bit-rate case also suffers from instability of performance improvement. We conjecture that this is due to a known problem in closed-loop design of predictive systems (see [10]), wherein there is a potentially considerable mismatch in statistics observed prior to parameter updates and after the update is made. We are currently working on adopting the asymptotic closed-loop design approach proposed in [10] to address this problem.

The performance gains for the test set, in terms of BD rate reduction, is summarized in Table 1, and the RD performance comparison for one of the test sequences is shown in Fig. 5. We can observe from these results that the trained coefficients provide significant performance improvement for video sequences with complex motion, as the proposed approach captures this by accounting for all neighboring motion vectors, in contrast to the conventional BMC, which is restricted to employ some compromise approximation of complex motions within a block. Moreover, for such sequences, we also obtain larger gains for doing motion refinement as this improves taking neighboring motion vectors into account, even with the limited range of motion refinement. It is also worth noting that the motion compensated residuals are smoother due to reduced blockiness by interpolating multiple predictions, which also leads to rate savings in transform coding.

Table 1: BD rate reduction for the proposed approaches relative to VP9, evaluated outside the training sets.

Sequence	Without motion refinement	With motion refinement
Foreman	11.174	11.316
Bus	13.783	14.455
Ice	6.213	6.863
HighWay	9.500	9.969
BlowingBubbles	6.898	7.422
BQMall	7.804	7.891
Vidyo4	3.973	4.011
CrowdRun	9.068	9.266
BasketBallDrive	7.746	7.937
Average	8.462	8.792

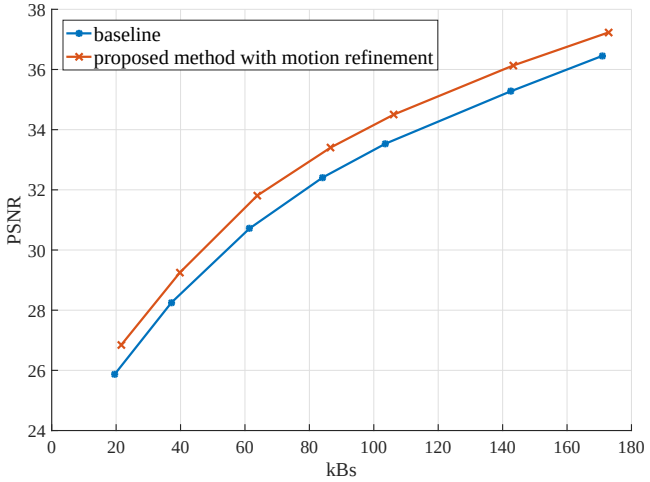


Fig. 5: RD performance comparison for the sequence *bus*.

5. CONCLUSION

A method for interpolating multiple motion compensated predictions with a choice of K modes is proposed to fully exploit the motion field and adapt to local statistics. Experimental results show that the proposed scheme can achieve an average 8.46% bit-rate saving over conventional fixed-block motion compensation method, and demonstrate the potential benefits of using nearby motion vectors to generate final predictions. Further improvement is expected with stable design of coefficients in low bit-rate regions, online adaptation of weights to currently observed statistics, and fully accounting for context while encoding the mode index. Future work will also include RD optimal mode selection and generalization to variable block sizes.

6. REFERENCES

[1] G. J. Sullivan, J. R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[2] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu, "A technical overview of vp9 - the latest open-source video codec," *SMPTE Motion Imaging Journal*, vol. 124, no. 1, pp. 44–54, 2015.

[3] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Transactions on Image Processing*, vol. 3, no. 3, pp. 327–331, 1994.

[4] H. Huang, J. W. Woods, Y. Zhao, and H. Bai, "Control-point representation and differential coding affine-motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 10, pp. 1651–1660, 2013.

[5] T. Wedi, "Adaptive interpolation filters and high-resolution displacements for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp. 484–491, 2006.

[6] H. Watanabe and S. Singhal, "Windowed motion compensation," *in Proc. of the SPIE Conf. on Visual Communications and Image Processing*, pp. 582–589, 1991.

[7] S. Nogaki and M. Ohta, "An overlapped block motion compensation for high quality motion picture coding," *in Proc. IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 184–187, 1992.

[8] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 693–699, 1994.

[9] Y.-W. Chen, T.-W. Wang, Y.-C. Tseng, W.-H. Peng, and S.-Y. Lee, "A parametric window design for obmc with variable block size motion estimates," *in Proc. Int. Workshop Multimedia Signal Processing*, 2009.

[10] H. Khalil, K. Rose, and S. L. Regunathan, "The asymptotic closed-loop approach to predictive vector quantizer design with application in video coding," *IEEE transactions on image processing*, vol. 10, no. 1, pp. 15–23, 2001.