Adaptive Interpolated Motion-Compensated Prediction with Variable Block Partitioning

Wei-Ting Lin, Tejaswi Nanjundaswamy, Kenneth Rose

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

Abstract

Conventional video coders rely heavily on pixel-domain block matching to remove temporal redundancies. This prediction structure constrains pixels within a block to use the same motion vector, which is ineffective for blocks with complex motion. To mitigate this shortcoming we recently proposed a new paradigm of adaptive interpolated motion compensation (AIMC), wherein neighboring motion vectors are considered as pointers to multiple estimation sources, which are linearly combined to form the final prediction, with weights chosen from pre-trained K-sets to capture variations in statistics. While promising initial results were obtained for fixed block sizes, this paper extends the approach to the important setting of variable block size partitioning, which has become standard in state-of-the-art video coding. Specifically, we propose a non-trivial generalization of AIMC to account for arbitrary block partitioning by "virtually" breaking a block to match its non-causal neighbors, and creating an interpolation tree structure, whose nodes extend from the original partitioning. This provides multiple estimates at the leaf nodes of the tree and enables an effective AIMC implementation. Experimental results validate the proposed paradigm with significant bit rate savings over conventional motion compensated prediction.

1 Introduction

In current video coders, such as AV1 [1] [2] and HEVC [3], inter-prediction is the main tool for exploiting temporal dependencies. Instead of encoding raw pixel values, the encoder divides a frame into non-overlapping blocks, and predicts each block by searching for a similar block in one or more previously reconstructed frames. This one-to-one pixel domain block matching is usually referred to as block-based motion compensation (BMC), and the difference in position between target and reference block is specified by a motion vector. BMC implicitly imposes a pure translation assumption, i.e. all pixels in a block move uniformly. It is thus obvious that BMC can cause significant prediction errors when a block's motion is non-translational, as in the case of zoom or rotation, and when the block overlaps multiple objects with different motions.

To account for more complex motion, reflected in the underlying motion field, the paradigm of adaptive interpolated motion compensation (AIMC) [4] was proposed. The main idea of AIMC is to explicitly treat the neighboring motion vectors as pointers to multiple observation sources for estimating a pixel in the current block. The final prediction is constructed by using optimal linear estimation coefficients to combine these sources. As a result, AIMC breaks free from BMC's limitation, namely,

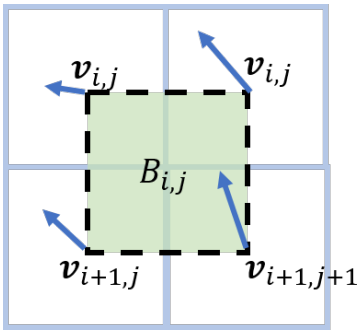
restricting a motion vector's influence to apply only and uniformly within a given block in a rigid rectangular block structure. Related ideas had previously led to the overlapped block motion compensation (OBMC) approaches [5–7]. However, OBMC approaches use a single type of extended window to effectively average overlapping observations. In distinction with OBMC, the AIMC approach focuses on implementing the optimal linear predictor for each pixel from observations obtained through multiple nearby motion vectors. Furthermore, AIMC designs K-sets of estimation coefficients that are trained to capture variation in local statistics. The predictor adapts to the content by switching between K-sets of coefficients. Specifically, AIMC can adapt in terms of how the estimates due to neighboring motion vectors should be weighted, and can further account for arbitrary object shapes.

The AIMC approach provided substantial gains under fixed block size settings [4]. This paper extends it to accommodate variable block size motion compensation (VBMC), on which all recent coding standards rely heavily for flexible partitioning [8]. VBMC poses a significant challenge on AIMC implementation, as the AIMC approach operates on off-grid blocks that lie between motion vectors. Since VBMC results in an unevenly spaced motion grid, the shapes of such off-grid blocks are not necessarily rectangular, and the number of possible off-grid block shapes is large. Hence, it would be impractical to train and store prediction coefficients for all possible shapes of off-grid blocks.

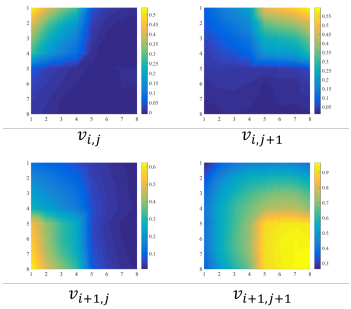
We propose to circumvent this difficulty via a generalization of the AIMC approach to accommodate the variable block size setting. This is achieved by "virtually" (i.e., only for the purpose of interpolation) breaking a block to match the minimum block size of its non-causal neighbors and creating an appropriate interpolation tree structure, wherein each node extends from the original block partitioning. With the aid of this tree structure, only square interpolation blocks of a few possible sizes are formed. Therefore, we can train $K-$sets of prediction coefficients for each possible size, and apply AIMC to each node of the interpolation tree. As the interpolation tree can be inferred from the original partition structure available to the decoder, no additional side information is needed. The choice of coefficient set needs to be transmitted, similar to fixed block AIMC. In general AIMC is more beneficial to locations exhibiting complex motion, whereas for nearly static regions, the additional side information might outweigh the benefit of improvement in prediction accuracy. Therefore, we incorporate a flag per superblock/CTU to indicate whether AIMC is enabled, so that the encoder can optimally decide to spend the side information only when it is beneficial. The experimental results validate the proposed paradigm with more than 5% bit-rate savings for video sequences with complex motion, and an average 2.35% bit-rate savings when compared to the conventional VBMC.

2 Background

This section briefly introduces AIMC, as the objective of this paper is to generalize AIMC to the variable block size setting. A more detailed description of AIMC can be found in [4]. Consider conventional motion compensated predition, which can be



(a) Grid of motion vectors.



(b) An example set of coefficient distributions for the corresponding motion vectors.

Figure 1: Fixed block size adaptive interpolated motion compensated prediction.

written as,

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

where, $\mathbf{v}_{i,j}$ denotes the motion vector for the (i,j) block, and $\hat{x}_{k-1}(\cdot)$ is a reconstructed pixel in the previous frame. Since a single motion vector cannot accurately capture complex motions within a block, the AIMC approach utilizes nearby motion vectors to obtain additional estimates which can be combined in a way that varies across the block. AIMC defines a block, $B_{i,j}$, lying between the 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. 1a. Since a motion vector captures the average motion for a standard block, we assume the effective position of a motion vector to be at the center of its standard block. Thus the AIMC block as defined is off-grid, covering one quadrant each of four neighboring standard blocks.

The final prediction for the AIMC off-grid block $B_{i,j}$ is generated by linearly combining the estimations with appropriate coefficients. Specifically, let $\mathbf{s}^{tl}_{i,j}$ be the top-left pixel in $B_{i,j}$, and $\mathbf{s}' = \mathbf{s} - \mathbf{s}^{tl}_{i,j}$, be the relative position within the block. The predictor for each pixel $\mathbf{s} \in B_{i,j}$ in frame k is

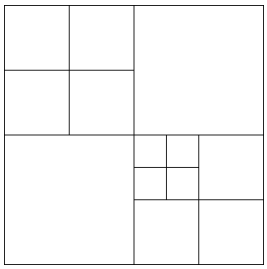
$$\tilde{x}_k(\mathbf{s}) = \sum_{m=0}^{1}\sum_{n=0}^{1} c^q_{m,n}(\mathbf{s}')\hat{x}_{k-1}(\mathbf{s} - \mathbf{v}_{i+m,j+n}) \tag{2}$$

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

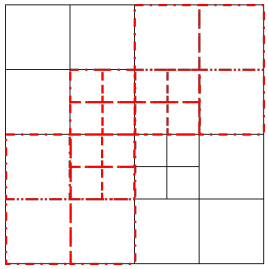
where $c^q_{m,n}(\mathbf{s}')$ is the q-th set's coefficient for prediction at position $\mathbf{s}'$ using the corresponding neighboring motion vector. The set of coefficients is selected to minimize the mean squared prediction error (MSE),

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

Fig. 1b provides an example set of coefficients. As discussed in Sec. 1, the coefficients allow us to adapt to the local statistics and capture significance of estimates due to neighboring motion vectors. The pre-defined K-sets of coefficients are stored in both encoder and decoder, and only the index of the selected set needs to be signaled to the decoder.

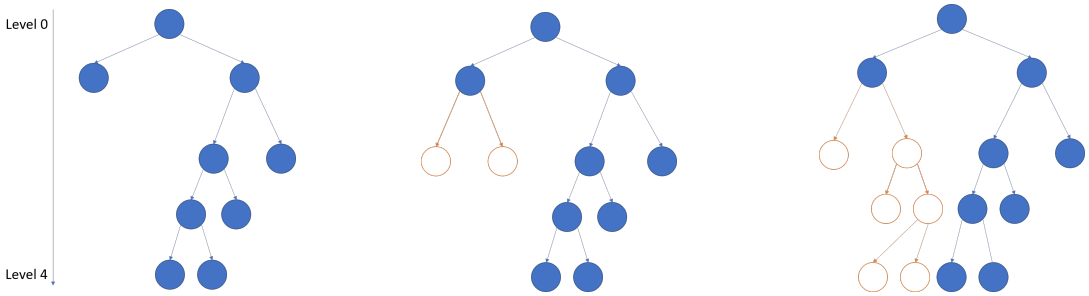


(a) Original Partition.



(b) Interpolated Block Partition.

Figure 2: An example of original partition and the inferred interpolated block partition.



(a) Original Partition.

(b) Level 2 Partition Matching.

(c) Final Partition.

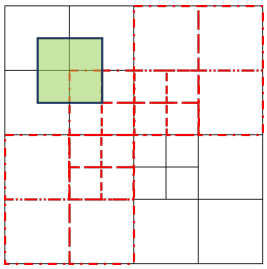
Figure 3: An 1-D example for block size matching algorithm.

3 Generalization of Adaptive Interpolation in Variable Block Size Coding

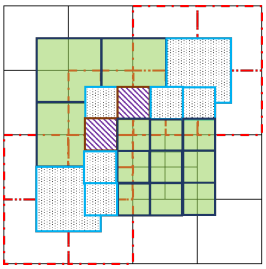
In the fixed block setting AIMC defines an off-grid block within the motion vector grid to exploit nearby motion vectors, and form an interpolated prediction for the off-grid block. However, in the variable block size setting, different block sizes could be employed resulting in unevenly spaced motion grid. This dramatically increases the number of the possible off-grid block patterns. Training and storing K-sets of coefficients for each possible pattern would be impractical.

To overcome this challenge, we propose an interpolation tree structure that can account for arbitrary block partitioning while still maintaining the simplicity of the fixed block size setting. Specifically, we propose to "virtually" break the blocks to match the non-causal neighbors' block sizes to provide interpolation to a smaller neighboring block. An example of such a division is illustrated in Fig. 2. This required partition for interpolated prediction can be inferred from the original partition. Therefore, no side information is needed for this new partition structure.

An illustrative 1-D example (with binary tree instead of quad-tree for 2-D) is shown in Fig. 3 to demonstrate the construction process for a interpolated block partition. Starting from the root (level 0), at each level, we split a node to match the partition of its non-causal neighbors (the right neighbor in the 1-D case). Explicitly,



(a) Interpolated prediction built for the top-right block.



(b) Interpolated prediction for the entire partition structure

Figure 4: Interpolated prediction built for the example partition in Fig. 2

let *root* be the root of the structure. The non-causal block partition matching can be completed via the function PARTITION_MATCHING(*root*) described in Algorithm 1, wherein the function *non_causal_neighbor_is_split* returns true if at least one of the non-causal neighbors of the current block is divided into smaller blocks in the original partition structure. We note that the original partition structure can be constructed by reading the *node.is_split* field and the new partition structure is constructed by the $node.force\_split$ field. In the 2-D quad-tree structure, we define the non-causal neighbors to be the right, bottom, or bottom-right of the current block, and we treat rectangular blocks as a combination of two smaller square blocks (therefore, all the blocks in the interpolated tree structure are squared).

Algorithm 1 Non-causal Block Partition Matching

```
function PARTITION_MATCHING(node)
    if node is empty then return
    end if
    need_to_split ← non_causal_neighbor_is_split(node)
    node.force_split = node.is_split || need_to_split
    for each child in node.children do
        PARTITION_MATCHING(child)
    end for
end function
```

We perform this block matching algorithm for each superblock/CTU, and shift each block towards bottom-right corner by half of the block size (we ignore the blocks shifted outside the superblock/CTU boundaries). We refer this new partition structure as interpolation tree. Each node in an interpolation tree is an off-grid block with respect to the original partitioning, and each quadrant of a off-grid block only cover one standard BMC block. Therefore, there is no ambiguity in assigning motion vectors to an off-grid block. A standard block's motion vector is assigned to the overlapped off-grid block's corner, and the AIMC approach is applied to each off-grid block. Fig. 4 illustrates how interpolated predictions are done for the example partition in Fig. 2. Breaking the blocks using the proposed algorithm results in very limited off-grid block patterns, leading to reduction of complexity and simplification

(a) The reconstructed frame.

(b) Interpolated prediction area of the corresponding frame shown in Fig. 5a.

Figure 5: The reconstructed frame and the corresponding interpolated prediction area (colored with black).

of implementation. As a result, we can design $K-$sets of estimation coefficients for each possible off-grid block pattern using the training algorithm similar to the original work in [4].

Note that there are three types of interpolated blocks in this structure. First, the blocks lying across the original block partition boundaries are valid blocks (solid blocks in Fig. 4), where AIMC can be performed. Second, the blocks lying entirely within the original blocks are non-valid interpolation blocks (dotted blocks in Fig. 4). No interpolation is done for this kind of blocks and side information is saved. Finally, the blocks which generate predictions that overlap to some extent with previously predicted regions (diagonal-striped blocks in Fig. 4). By construction, only a part of large interpolated block can be over-written by smaller interpolation blocks. Therefore, we allow new interpolated predictions to overwrite the old ones based on the scan order to refine the interpolated prediction.

3.1 Deployment of Interpolated Prediction

Clearly, an area with complex texture and motion benefits most from the interpolated prediction framework, even at the cost of transmitting the interpolation modes. For the area that can be well predicted by the conventional prediction, the additional side information will result in degradation of RD performance. Therefore, we add a flag to each superblock/CTU to indicate whether the proposed interpolated prediction is used for a superblock/CTU. The RD costs of both original and interpolated predictions of a superblock/CTU are computed. The flag is on if the RD cost of the interpolated prediction is less than the RD cost of the original prediction; otherwise the flag is off and interpolated prediction will not be used for the entire superblock/CTU. Fig. 5 shows a reconstruction frame in the *Flower* sequence, and the corresponding area where interpolated prediction is applied. The proposed on-off mechanism allows the coder to refine the prediction only when it is necessary.

4 Experimental Results

We evaluate the proposed approach using the AV1 framework [1]. It is important to emphasize that the proposed paradigm can be applied to any modern coders as they all employ variants of VBMC. For simplicity of simulations, we restrict the minimum partition block size to be 8×8, which limits the valid interpolated prediction block sizes to be 8×8, 16×16 and 32×32. We also do not allow intra and compound predictions for inter-frames and only use intra prediction to code the first frame of each sequence. The coefficients of all block sizes are initialized using 2-D raised cosine function, which is given 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 with a parameter B equal to the corresponding block size,

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

and $C(x, y)$ is the normalization function.

We select $(\beta_x, \beta_y) \in \{(0, 1), (1, 0), (1, 1)\}$ for 8×8 and 16×16 blocks, and $(\beta_x, \beta_y) \in \{(0, 0.5), (0.5, 0), (0.5, 0.5)\}$ for 32×32 blocks (i.e $K = 3$ for all sizes of blocks), and the initial coefficients are generated by uniformly sampling the function $H_{2D}(\beta_x, \beta_y, x, y)$ for $0 \leq x, y \leq 1$. The coefficients are trained using the algorithm described in the previous work [4] in fixed block size setting with QP equals to 20. The same sets of trained coefficients are then applied to all the videos coded under the variable block size setting in different target bit rate region and different range of resolution. The training set contains first 100 frames of *Flower*, *BlowingBubbles*, *Freman*, and *Coastguard* sequences. The coefficients are stored in both encoder and decoder side. The mode index and the interpolated prediction enabled flag are written into bitstream by using the symbol writer in the AV1 codec.

For testing, we encode the first 100 frames of each video sequences using target bit rate mode, wherein the coder can adjust QP values to match a given bit-rate. The bit-rate range is selected to cover a wide range of qualities as shown in Fig. 6. The performance gain in terms of BD rate reduction is summarized in Table 1. From the results, we observe that for the sequences with nearly static scenes, such as *Bridge-far* and *Bridge-close*, the proposed paradigm provides little gains since VBMC already provides decent predictions, and the prediction quality improvement of the proposed approach cannot compensate for the rate cost. Therefore, most of the superblocks in these sequences have interpolated prediction disabled. However, for videos with complex motion, the interpolated blocks and the trained coefficients properly account for the neighboring motion vectors, and therefore provide substantial gains. By contrast, the conventional VBMC is restricted to use a compromised single motion to approximate complex motions and predict the entire rigid rectangular block.

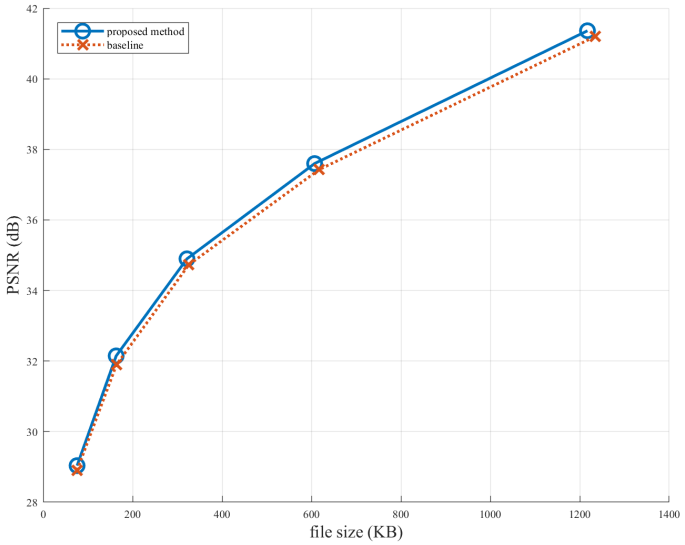


Figure 6: Coding performance comparision for sequence *Mobile*.

Table 1: BD rate reduction for the proposed approaches relative to AV1

Sequence	Bit-Rate Reduction (%)
Waterfall	4.371
Flower	5.874
Stefan	2.977
Mobile	5.106
Coastguard	0.486
Bridge-close	0.516
Bridge-far	0.401
Foreman	1.395
Bus	2.824
Akyio	1.075
BlowingBubbles	2.035
BQSquare	4.749
BQMall	1.795
Shields	1.313
Park_run	1.069
Average	2.3505

5 Conclusion

In this paper, a novel approach is proposed to generalize the AIMC method to account for the variable block size setting. This approach enables substantial gains from em-

ploying variable block sizes partitioning, and further improve its performance while maintaining simplicity of the original AIMC approach. A flag is added per-superblock to adapt to the content of video frame to properly enable the interpolated prediction when necessary. The experimental results show that the proposed approach significantly improves the RD performance. Further improvement is expected with optimal design of number of modes for each block size and online adaptation of coefficients based on hyper-local statistics. Future work will also include mode prediction to reduce side information.

References

[1] "AOM - Alliance for Open Media," http://aomedia.org/.

[2] D. Mukherjee, H. Su, J. Bankoski, A. Converse, J. Han, Z. Liu, and Y. Xu, "An overview of new video coding tools under consideration for vp10 the successor to vp9," in *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics, 2015, pp. 95 991E–95 991E.

[3] G. J. Sullivan, J. 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.

[4] W.-T. Lin, T. Nanjundaswamy, and K. Rose, "Adaptive interpolated motion compensated prediction," *in Proc. IEEE International Conference on Image Processing*, pp. 943–947, 2017.

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

[6] 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.

[7] 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.

[8] 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.