University of California
Santa Barbara

# Toward the High Precision Predictive Coding in Video Compression

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Wei-Ting Lin

Committee in charge:

Professor Kenneth Rose, Chair
Professor B. S. Manjunath
Professor Shiv Chandrasekaran
Doctor Debargha Mukherjee

September 2019

The Dissertation of Wei-Ting Lin is approved.

---

Professor B. S. Manjunath

---

Professor Shiv Chandrasekaran

---

Doctor Debargha Mukherjee

---

Professor Kenneth Rose, Committee Chair

July 2019

Toward the High Precision Predictive Coding in Video Compression



by

Wei-Ting Lin

*To my parents*

## Acknowledgements

I would firstly like to express my sincerest thanks to my advisor, Professor Kenneth Rose, for his guidance and support throughout my Ph.D. study. His insightful thoughts shape my perspectives and inspire me to tackle problems in an elegant way. Working with him was truly a pleasure and invaluable experience.

I am very thankful to Professor Chandrasekaran, Professor Manjunath, and Dr. Mukherjee, for serving on my doctoral committee and reviewing my work. Thanks to Ms. Val De Veyra, for her consistent support to all graduate students.

This research has been funded by Google Inc., where I have the honor to work with the video codec team for my summer internships. Special thanks to Dr. Debargha Murherjee and Dr. Yue Chen, from whom I learn a lot in both programming and doing research in industry.

I thank all my labmates in SCL for making my stay here joyful. I thank Dr. Yue Chen for her help during my initial time at SCL, and Dr. Tejaswi Nanjundaswamy for the interesting discussions to formulate this work at its early stage and for polishing my funny writing. I thank Dr. Bohan Li; together, we had a fun time struggling to implement algorithms into video codec. It was a pleasure to collaborate with all of them. My appreciation also goes to other members in SCL: Bharath, Clint, Sina, Shunyao, Ahmed, Mahmoud and Zeyu. I am grateful to all my friends here in Santa Barbara, across Unite States, and back in Taiwan for all their support and companionship through good times and bad during my

Ph.D. study.

Finally, I would like to express my deepest gratitude to my parents for their unconditional love and support. The completion of this thesis would not have been possible without them.

# Curriculum Vitæ

Wei-Ting Lin

## Education

| | |
|---|---|
| 2012 | Master of Science in Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan |
| 2010 | Bachelor of Science in Electronic Engineering, National Taiwan University, Taipei, Taiwan |

## Professional Experience

| | |
|---|---|
| 06/2015 – 08/2019 | Graduate Student Researcher,<br>Dept. of Electrical and Computer Engineering, University of California, Santa Barbara |
| 06/2018 - 09/2018 | Summer Intern,<br>Chrome Media, Google Inc., Mountain View, CA |
| 06/2017 - 09/2017 | Summer Intern,<br>Chrome Media, Google Inc., Mountain View, CA |
| 06/2016 - 09/2016 | Summer Intern,<br>Chrome Media, Google Inc., Mountain View, CA |
| 06/2015 - 09/2015 | Summer Intern,<br>Advanced Communication Tech., MediaTek Inc., San Diego, CA |

## Publications

Lin, Wei-Ting, Bohan Li, and Kenneth Rose. "Multi-hypothesis Predictive Coding in Video Compression." to be submitted to IEEE Transactions on Image Processing

Lin, Wei-Ting, Tejaswi Nanjundaswamy, and Kenneth Rose. "Adaptive Interpolated Motion-Compensated Prediction with Variable Block Partitioning." In 2018 Data Compression Conference, pp. 23-31. IEEE, 2018.

Lin, Wei-Ting, Zoe Liu, Debargha Mukherjee, Jingning Han, Paul Wilkins, Yaowu Xu, and Kenneth Rose. "Efficient AV1 video coding using a multi-layer framework." In 2018 Data Compression Conference, pp. 365-373. IEEE, 2018.

Lin, Wei-Ting, Tejaswi Nanjundaswamy, and Kenneth Rose. "Adaptive interpolated motion compensated prediction." In 2017 IEEE International Conference on Image Processing (ICIP), pp. 943-947. IEEE, 2017.

Liu, Zoe, Debargha Mukherjee, Wei-Ting Lin, Paul Wilkins, Jingning Han, and Yaowu Xu. "Adaptive multi-reference prediction using a symmetric framework." Electronic Imaging 2017, no. 2 (2017): 65-72.

**Abstract**

Toward the High Precision Predictive Coding in Video Compression

by

Wei-Ting Lin

The main focus of this dissertation is on the optimal design of motion compensation scheme for predictive coding in video compression. The "sub-optimality" of conventional block-based motion compensation scheme, wherein the impact of a motion vector is confined within in a rigid rectangular block, motivates our design of a multi-hypothesis motion compensation scheme. We explicitly treat motion vectors as pointers to observation sources. Given the high-correlation between adjacent pixels in nature images, the motion vectors of neighboring blocks can point to relevant estimates of the current target. This dissertation work builds on this paradigm and demonstrates advanced techniques devised by incorporating the information of the entire motion vector field.

We first directly formulate the problem of motion compensation with multiple estimates as a linear estimation problem, and design a training method to derive the optimal linear coefficients to avoid overfitting as well as to minimize the ultimate reconstruction errors rather than prediction errors. As a single set of coefficients cannot capture the varying statistics of video sequences, we design $K$ sets of coefficients which are trained off-line through "K-mode" iterative clus-

tering techniques. By switching between the predefined sets of coefficients, the encoder can adapt to local statistics. This approach is then extended to the setting of variable block size partitioning to enjoy the substantial gain provided by the flexibility of dividing blocks to approximate object shapes. As the additional side information to indicate the set of prediction coefficients used to generate the final prediction is generally not negligible, a parametric framework is proposed to model the statistics of estimates and target pixels. The model leverages the first-order Markov property for image signals and relationships between motion vectors in the motion field. As a result, the coefficients derived from the model can automatically adapt to local variations without additional side information. Moreover, using the parametric approach, we can combine estimates from any number of motion vectors at any pixel location, which allows us to completely break free from the block structure by allowing a motion vectors influence to be of arbitrary shape.

The reminder of this dissertation is focused on optimization on the AV1 encoder, the open source video encoder founded by the Alliance of Open Media (AOM). We first introduce a new coding tool to extend the number of reference frames, and then we use the existing coding tools to design a coding structure, wherein the reference frames are allocated to cover a wider temporal range to offer more diversities. This new design allows the encoder to better capture temporal variations. Finally, we introduce a complementary compound prediction mode,

which is designed and optimized for the blocks where the existing compound modes fall short. Simulations provide experimental evidences for the efficiency of proposed mode with consistent coding gain across all bit-rate range.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Over the years, video streaming and content delivery services have become an essential part of our lives. Video has dominated the global internet traffic, and it will continue to represent 80 to 90 percent of total traffic at the end of 2022, according to the Cisco Visual Networking Index [1]. Moreover, the growing demand for fast and ultra high-quality videos put unaffordable loads upon networks as well as storage systems. These pressures motivate the development of a more efficient video compression algorithm, so as to provide higher quality of reconstructed videos with lower transmission bit-rate. The research in video compression emphasizes mainly on the three consecutive modules, predictive coding, transform coding and entropy coding. These three components complementarily exploit different types of redundancies of natural video signals to achieve better compression performance. In this dissertation, we focus on the predictive component and provide

estimation-theoretic approaches to exploit correlations between different predictors of individual pixels. The proposed paradigm offer more accurate predictions and thereby improvement in compression performance.

## 1.1 Motivation and Contributions

The current video coders, such as WebM [2] [3] and HEVC [4] [5], rely heavily on block-based motion compensation (BMC) to remove the temporal redundancy within video sequences. These coders divide a frame into non-overlapped blocks, and predict each block through pixel domain block matching using one or more previously reconstructed frames. The displacement between the target and reference blocks is referred to as a motion vector. This approach is an attractive solution to current codecs as it balances between the prediction accuracy and the overhead of both searching and signaling motion vector at each pixel. However, this simplification imposes a pure translation assumption on the pixels within a block (all pixels in a block move uniformly). It is thus obvious this approach cannot account for complicated motions such as zoom or rotation, and the scenario wherein a block contains multiple objects with different motions. Therefore, the major motivation of this dissertation is to develop a prediction scheme to effectively address this issue.

The main line of the research focuses on the optimality of inter-prediction. Specifically, we propose to explicitly treat each motion vector as a pointer to a

source of observation. This point of view opens a door to incorporate motion vectors from nearby blocks to estimate pixels as they can point to relevant observations due to the high correlations between adjacent pixels in nature images. The final prediction is constructed by weighting the observation properly according the influences of all available motion vectors to the pixel.

Inspired by the estimation-theoretic prediction framework, we propose the adaptive interpolation motion compensation (AIMC) method, wherein we formulate the problem of combining multiple estimates to form the final prediction as a linear estimation problem, and derive the optimal coefficients through a training algorithm that minimizes the reconstruction errors as well as avoid over-fitting. Prediction coefficients are further adapted to local statistics by switching between predefined sets of coefficients, which are trained through a "$K-$modes" clustering to minimize the rate-distortion cost. We then extend the work to accommodate an important setting of variable block size partitioning which all recent codecs adopt. To effectively train and store the coefficients on the unevenly sample grids resulted in variable block size partitioning, we propose a non-trivial generalization by "virtually" break a block to match its non-causal neighbors and creating an interpolation tree structure, whose nodes extended from the original partitioning. Within this structure, only square interpolation blocks of a few possible sizes are formed at the leaf nodes. Therefore, we can effectively train $K$-sets of coefficients for each possible size, and apply interpolation on each node. Simulation results

demonstrate the effectiveness of the proposed AIMC approach with significant gain over convention variable block motion compensation (VBMC).

Having validated the benefits of the prediction paradigm of using multiple estimations in the basic setting of AIMC, which accounts for influences of limited numbers of motion vectors within an interpolation area, with significant performance improvements. We propose to completely break free from the block structure with a new design which can allow a motion vector's influence to be of arbitrary shape, and allow any number of motion vectors to have influence on each pixel. To achieve this, we propose to employ a parametric approach to obtain the interpolation coefficients to generate the final prediction from multiple observations. Under the assumption of a separable first order Markov model for the image signals, we propose a model to incorporate this and the correlations between motion vectors in the motion field generated at the encoder. The prediction coefficients derived from the model can automatically adapt to local statistics without requiring additional side information. Experimental results show further improvement over the original non-parametric approach.

Another contribution of this dissertation focuses on optimization on AV1 codec, the open-source video coder founded by a group of top tech leaders including Google. We first introduce a new coding tool, which extends the total number of reference frames to cover a wider temporal range. This allows the encoder to adapt better to temporal variation. Using existing coding tools in

AV1, we then introduce a multi-layer coding structure to diversify the positions of the reference frames, which allow us to better leverage the increased number of references frames. However, with the variety choices of reference frames, the original compound prediction modes, which weights the two predictions from different directions in predefined ways, cannot adapt properly to the local variations of pixel values along the motion trajectory. We propose to pose the weight design problem as a linear estimation problem and devise a new compound mode which allows the encoder to weight the predictions more flexibly. The predictions are combined with the optimal weights derived to balance between side-information and prediction gain to minimize the ultimate rate-distortion coding cost. Simulation results demonstrate efficiency of the proposed method with consistent coding gain over all bit-rate range.

## 1.2 Dissertation Organization

The rest of this dissertation are organized as follows. In Chapter 2 we introduce the fundamental idea of non-parametric adaptive interpolated motion compensation (AIMC), wherein the weight design problem is formulated as a linear estimation problem. We then present a training method to avoid overfitting and derive the weights to minimize the actual reconstruction error rather than prediction error.

In Chapter 3, we introduce a generalization of AIMC to account for variable

block partitioning with the aid of an interpolation tree structure inferred from the original partition structure. We then present a careful design for deploying the proposed approach on the codec, which allows the encoder to flexibly enable/disable AIMC based on the rate-distortion trade-off to minimize ultimate coding cost.

In Chapter 4, a parametric approach to derive optimal linear coefficients is presented to account for arbitrary number of motion vectors' influences to a pixel. The proposed model, aiming at integrating both the Markovian assumption for image signals and the correlations between motion vectors in the motion field is described.

In Chapter 5, we introduce a new coding tool to AV1 codec, which allows the encoder to deploy more reference frames to cover a wider temporal range. A multi-layer coding structure is then devised to leverage the increased number of reference frames to diversify the reference frame positions. Finally, we present a new compound prediction mode acts as a complementary role to the existing compound mode to exploit predictions of different qualities.

Chapter 6 concludes this dissertation and suggests possible directions for future research.

# Chapter 2

# Adaptive Interpolated Motion Compensated Prediction (AIMC)

In this chapter we introduce a novel motion compensation scheme, adaptive interpolated motion compensation (AIMC), based on using available neighboring motion vectors. Standard motion compensation relies on pixel domain matching within a rigid block, 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. Therefore, we propose to explicitly treat 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.

## 2.1 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 [4] and VP9 [6], 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.

Some remedies are proposed to solve these issues. Variable block size motion compensation (VBMC) is the most popular one among the solutions and is used in all recent video coding standards [2–5]. The prediction accuracy is enhanced by allowing iterative break-down of current block to approximate the object shapes [7]. However, VBMC still impose a pure translation assumption on each block. To capture motion other than simple translation, control grid interpolation methods [8] [9] are proposed to use higher order models to account for more complex motions. The motion vectors obtained from BMC are used as control points to construct smoothly varying motion vectors between them, which allows pixels in the same BMC block can have different motion vectors, and potentially capture the other types of motions such as zoom or rotation. However, this smooth motion field assumption cannot account for discontinuous motions resulting from objects moving in different directions and the method is simply not applicable when the adjacent motion vectors points to different reference frames, which thus rendering the approach unavailing. 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 [10], which yield errors, see [11].

The aforementioned approaches still try to assign a single motion vectors to each block and limit a motion vectors influence to be constant and within a rigid rectangular block structure. Instead of imposing such restrictions on motion vectors, we can explicitly treat a motion vector as a pointer to a source of observation. This point of view opens a door to incorporate motion vectors from nearby blocks to estimate pixels as they can point to relevant observations in the current block. The final prediction is constructed by using optimal linear coefficients to combine these observations. As a result, we can break free from the BMC's limitation, namely, restricting a motion vector's influence to apply only and uniformly within a block in a rigid rectangular block structure. Related idea had previously led to the overlapped block motion compensation (OBMC) approaches [12–14], which design a single type of extended window centered around a motion vector position to effective average overlapping observations. In distinction with OBMC, our approach focuses on the area that lies between motion vector positions, and implements the optimal linear predictor for each pixel in the *off-grid* area. Moreover, instead of just using a single window, which can hardly account for complex motions reflected in the underlying motion filed, we design $K$-sets of estimation coefficients that trained to capture variation in local statistics. Specifically, by switching between the $K$-sets of coefficients, the proposed adaptive interpolation motion compensation (AIMC) approach allows the predictor to adapt in terms

of how the estimates due to neighboring motion vectors should be weighted, and potentially account for arbitrary object shapes.

In this work the focus is on implementing the optimal linear estimator for each pixel from observations 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.

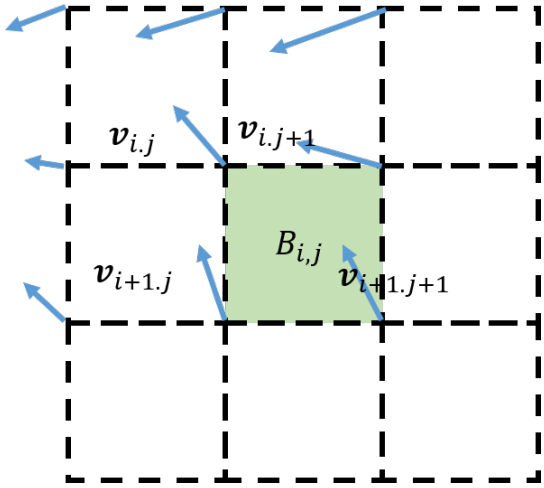



Figure 2.1: Grid of motion vectors.

## 2.2 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

$$\tilde{x}_k(\mathbf{s}) = \hat{x}_{k-1}(\mathbf{s} - \mathbf{v}_{i,j}), \tag{2.1}$$

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. 2.1. 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 $\mathbf{s}^{tl}_{i,j}$ 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 overall prediction for the pixel $\mathbf{s} \in B_{i,j}$ in frame $k$ is calculated as

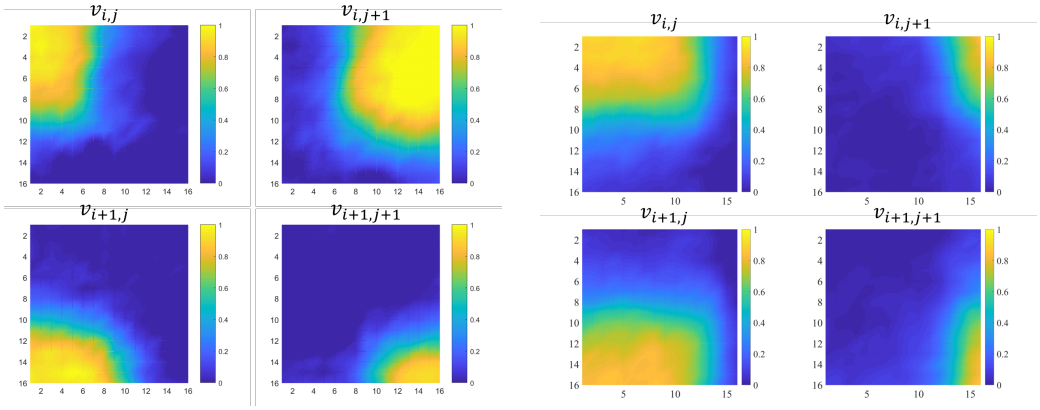
$$\tilde{x}_k(\mathbf{s}) = \sum_{m=0}^{1}\sum_{n=0}^{1} c^q_{m,n}(\mathbf{s}')\hat{x}_{k-1}(\mathbf{s} - \mathbf{v}_{i+m,j+n}) \tag{2.2}$$

$$= \mathbf{c}^q(\mathbf{s}')^\intercal \hat{\mathbf{x}}_{k-1}(\mathbf{s} - \mathbf{v}), \tag{2.3}$$

where $c^q_{m,n}(\mathbf{s}')$ is the $q$-th set coefficient for prediction at position $\mathbf{s}'$ using the corresponding $(m, n)$ neighboring motion vector. Equation (2.2) is shown in vector form in (2.3) where $^\intercal$ denotes transposition. The set of coefficients is selected to minimize the mean squared prediction error, i.e.,

$$q = \underset{r\in\{0,\ldots,K-1\}}{\arg\min} \sum_{\mathbf{s}\in B_{i,j}} \Big( x_k(\mathbf{s}) \; - \mathbf{c}^r(\mathbf{s}')^\intercal \hat{\mathbf{x}}_{k-1}(\mathbf{s} - \mathbf{v}) \Big)^2. \tag{2.4}$$

Two example sets of coefficients is shown in Fig. 2.2. The coefficients tend to approach one near the corresponding motion vector position and decrease with distance. As discussed in Sec. 2.1, applying coefficients this way allows us to




(a) An example sets of weight distributions. (b) An example sets of weight distributions.

Figure 2.2: Example sets of weight distributions corresponding to motion vectors $\mathbf{v}_{i,j}, \mathbf{v}_{i+1,j}, \mathbf{v}_{i,j+1}$ and $\mathbf{v}_{i+1,j+1}$

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.

## 2.3 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 objective 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.,

$$\hat{x}_k(\mathbf{s}) = \tilde{x}_k(\mathbf{s}) + \hat{e}_k(\mathbf{s}). \tag{2.5}$$

Designing coefficients to minimize prediction error leads to better prediction, but this does not guarantee better reconstruction. Moreover, the statistics of the block prediction errors using a single motion vector has a long tail as shown in Fig. 2.3. It is because even for the blocks that do not have good predictions in reference frame (such as occluded object), encoders will still try their best to find the most similar matches. This can result in large prediction errors to target blocks and they can strongly distort the classical least-squares estimator and results in poor generalization [15]. Hence, we propose to design the coefficients while accounting for the reconstruction error, which allow us to design weights directly leads to better reconstruction and also have the predictions with large errors be compensated by the prediction residuals. As shown in Fig. 2.4, by accounting prediction residuals, the reconstruction errors are confined in the certain range.

Once blocks are classified into $K$ clusters based on (2.4), the square reconstruction error for each cluster $C_q$ is

$$J = \sum_{B_{i,j} \in C_q} \sum_{\mathbf{s} \in B_{i,j}} \Big( x_k(\mathbf{s}) - \tilde{x}_k(\mathbf{s}) - \hat{e}_k(\mathbf{s}) \Big)^2. \tag{2.6}$$

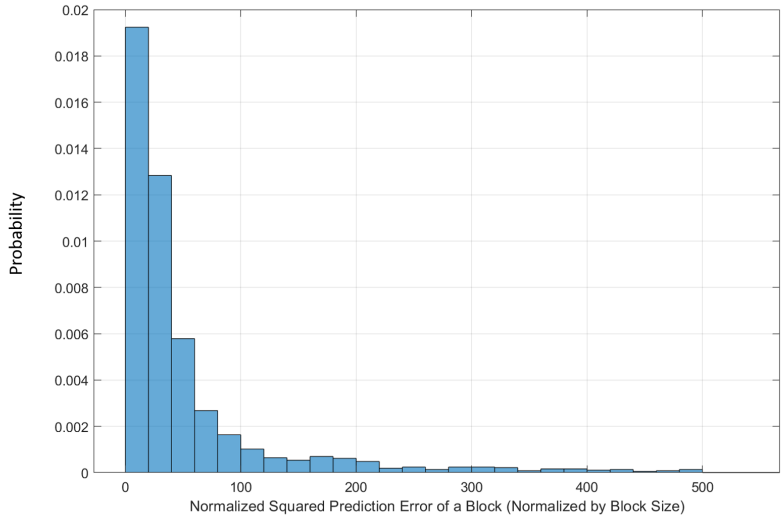
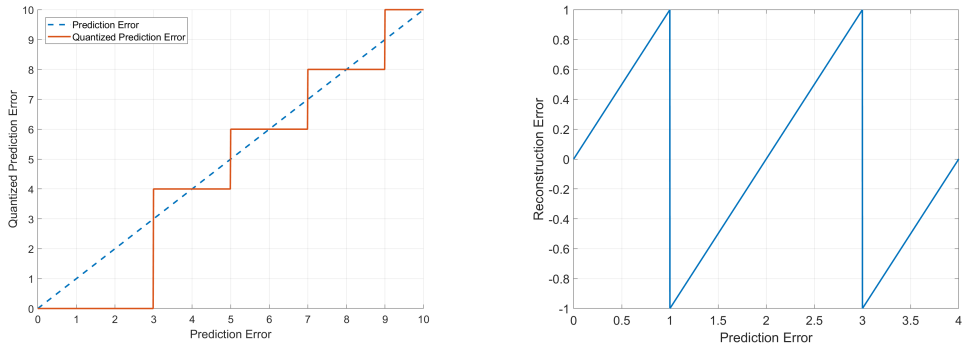



Figure 2.3: Statistic of squared prediction error of a block (normalized by the block size)




(a) Prediction error vs. Quantized prediction error.

(b) Prediction error vs. Reconstruction error.

Figure 2.4: An example of applying quantization to prediction residuals. (quantization step $q = 2$ and dead-zone $= 2$).

Given the discreet nature of quantization, this cost (2.6) 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 (2.5), hence optimal predictive coefficients $\mathbf{c}^q(\mathbf{s}')$ must satisfy

$$\mathbf{c}^q(\mathbf{s}') = E[\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})^\intercal]^{-1}E[\tilde{x}_k(\mathbf{s})\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})]. \tag{2.7}$$

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 (2.4).

2. Given the new training set, (2.7) is employed to compute the new coefficients.

Fig. 2.5 shows the average PSNR improvements at each iteration during training for different target bit-rate regions. In the low bit-rate case, most of the prediction residuals are quantized to zero. The improvement in prediction accuracy can directly map to improvement of reconstruction, and hence the significant improvement in the first few iterations is observed. However, in mid and high target bit-rate regions wherein the coefficients will change more slowly in the presence of quantized prediction error. The improvement is much smoother.

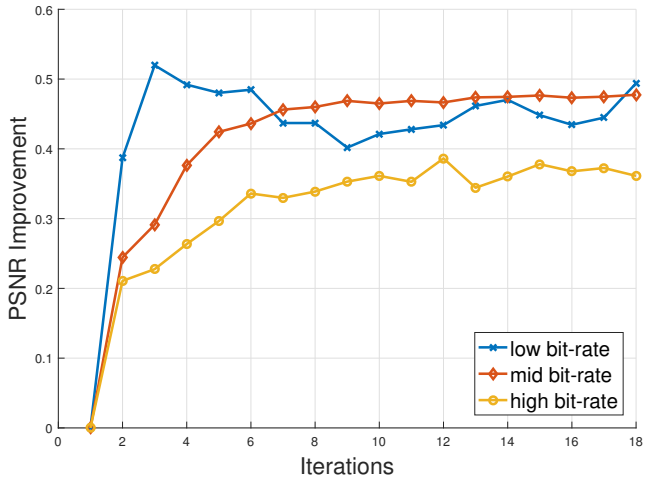



Figure 2.5: PSNR improvement (in dB) versus iterations of the proposed $K$-mode clustering algorithm for different target bit rate regions.

### 2.3.1 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

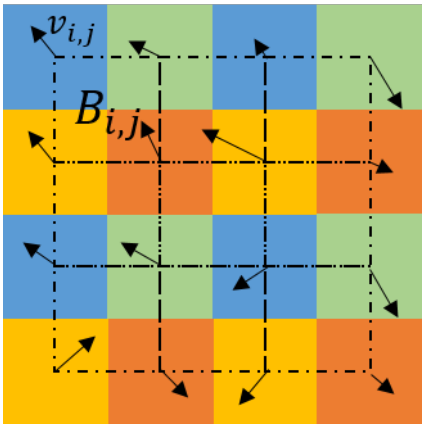



Figure 2.6: Motion vectors of blocks shown with the same color can be updated in parallel during motion refinement for interpolated prediction.

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. 2.6, all motion vectors shown with the same color can be updated in parallel.

## 2.4 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 $16 \times 16$ 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,

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

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

The performance gains for the test set, in terms of BD rate reduction, is summarized in Table 2.1, and the RD performance comparison for one of the test

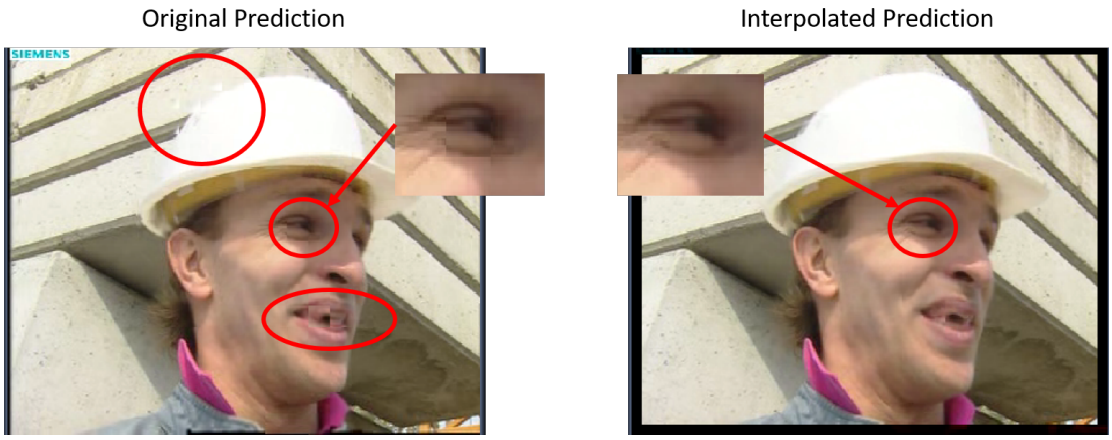



Figure 2.7: Inter-prediction Comparison.

sequences is shown in Fig. 2.8 and 2.9. 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 as shown in Fig. 2.7, which also leads to rate savings in transform coding.

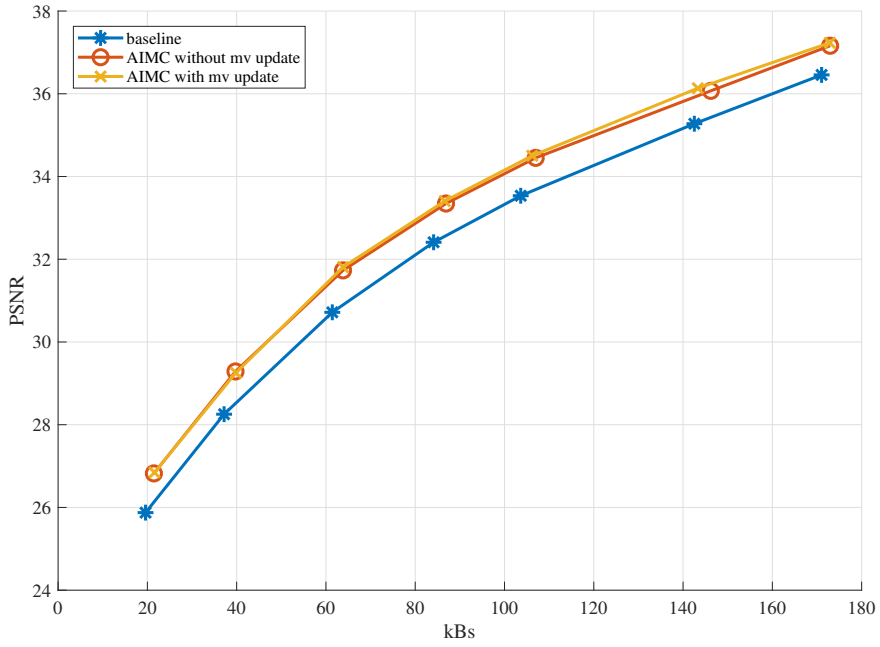



Figure 2.8: RD performance comparison for the sequence *bus*.

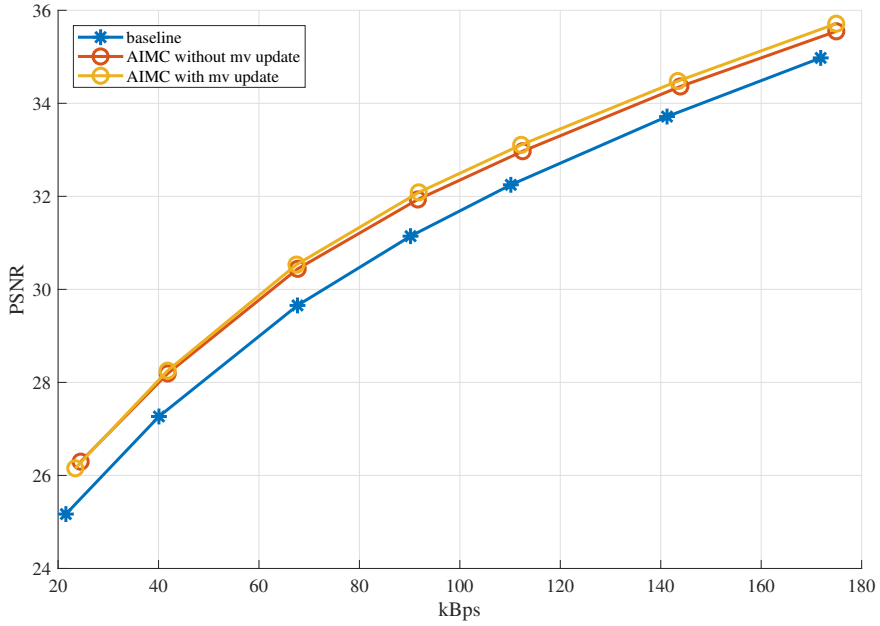



Figure 2.9: RD performance comparison for the sequence *flower*.

Table 2.1: 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 |

# Chapter 3

# AIMC with Variable Block Partitioning

To mitigate the shortcoming of BMC, in Chapter 2, we 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 to an off-grid block with weights chosen from pre-trained $K$-sets to capture variations in statistics. While promising initial results were obtained for fixed block sizes, in this chapter, we extend 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.

## 3.1 Introduction

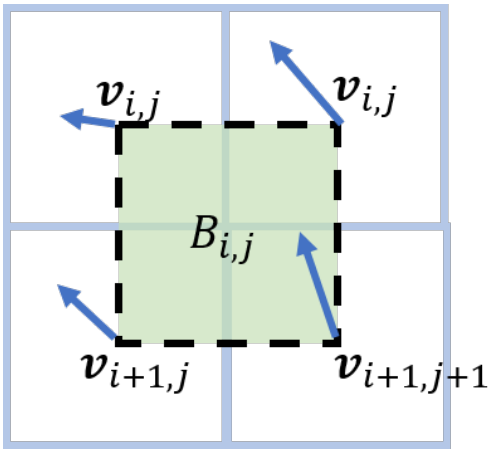
In modern video coders, block-based motion compensation (BMC) is a prevailing tool for removing temporal redundancies within a video sequence, as it achieves a good trade-off between the prediction accuracy and the overhead of searching and signaling motion vector at each pixel. The prediction accuracy of BMC is further improved by variable block motion compensation (VBMC), where the size of a block is allowed to vary in order to match the shape of an object. Even VBMC has become the-state-of-the-art and implemented within all current coders [2–4, 6], it is still inefficient to account for non-translational motions.

To account for more complex motion, reflected in the underlying motion field, the paradigm of adaptive interpolated motion compensation (AIMC) [16] was premised in Chapter 2. 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

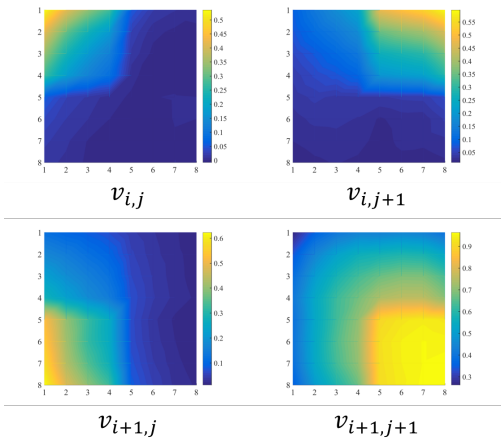
free from BMC's limitation, namely, restricting a motion vector's influence to apply only and uniformly within a given block in a rigid rectangular block structure. Related ideas had previously led to the overlapped block motion compensation (OBMC) approaches [12–14]. However, OBMC approaches use a single 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 [16]. In this chapter, we extend it to accommodate variable block size motion compensation (VBMC), on which all recent coding standards rely heavily for flexible partitioning [7]. 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 the 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 2.56% bit-rate savings when compared to the conventional VBMC.




(a) Grid of motion vectors.

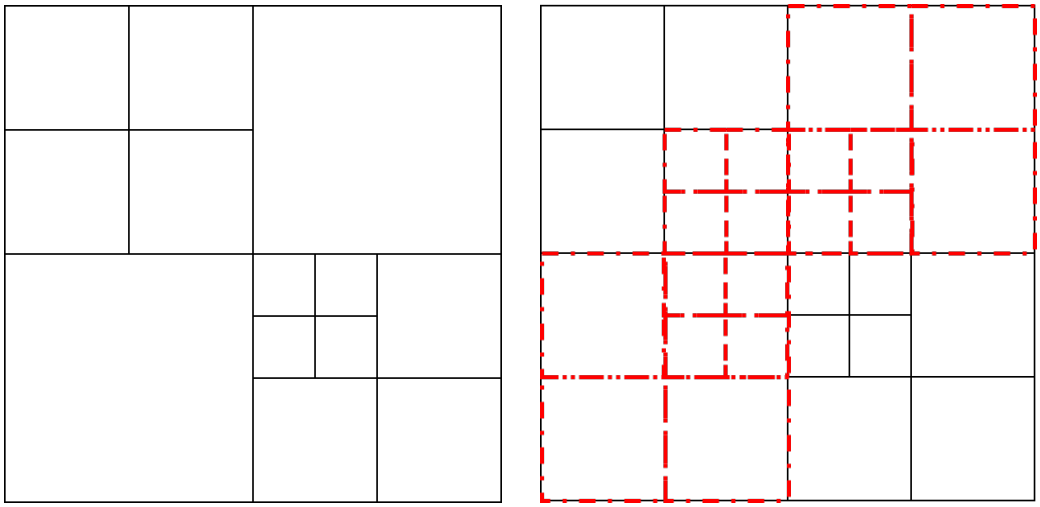



(b) An example set of coefficient distributions for the corresponding motion vectors.

Figure 3.1: Fixed block size adaptive interpolated motion compensated prediction.

## 3.2 Generalization of Adaptive Interpolation in Variable Block Size Coding

Recall in Chapter 2, we define an off-grid block between motion vectors, and generate multiple estimations from the motion vectors. The final prediction to the off-grid block is formed by linearly combining the estimations with appropriate coefficients, which trained offline as shown in 3.1. As discussed in Chapter 2, the different sets of 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. 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-



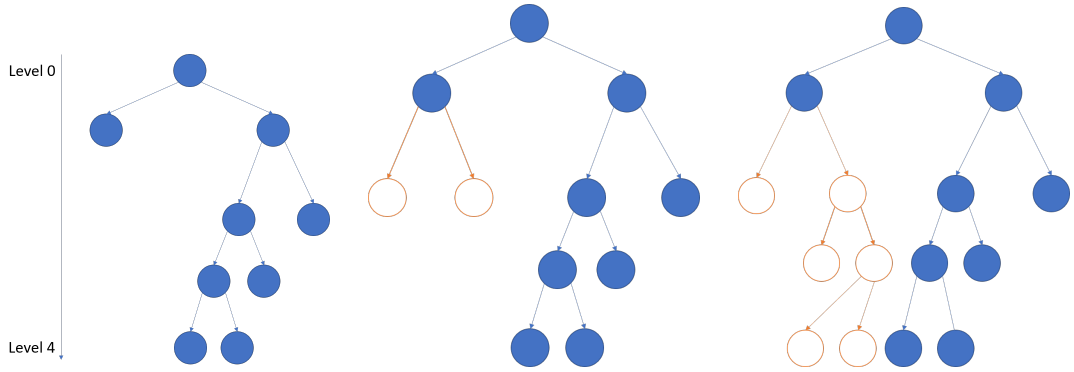
(a) Original Partition. (b) An example set of coefficient distributions for the corresponding motion vectors.

Figure 3.2: An example of original partition and the inferred interpolated block partition.

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. 3.2. 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. 3.3 to demonstrate the construction process for an interpolated block partition. Starting from the root (level 0), at each level, we split a node




(a) Original Partition. (b) Level 2 Partition Matching. (c) Final Partition.

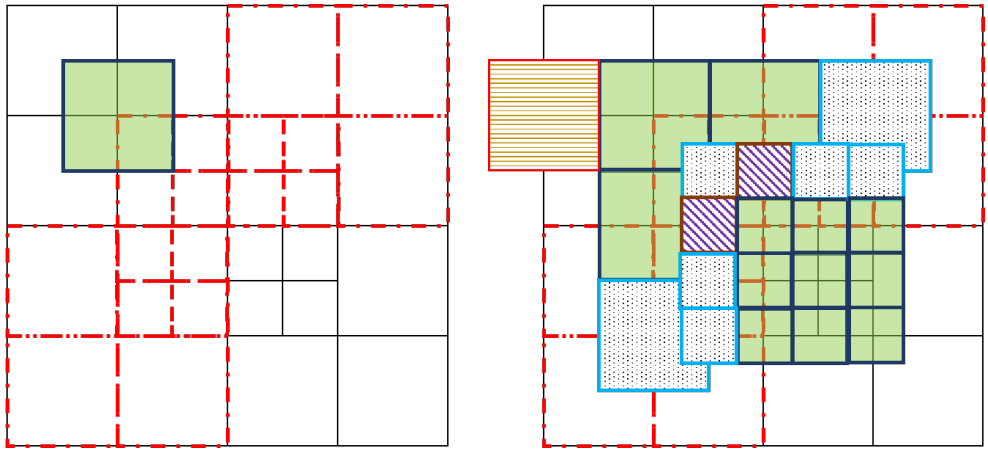
Figure 3.3: A 1-D example for block size matching algorithm.

to match the partition of its non-causal neighbors (the right neighbor in the 1-D case). Explicitly, 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).

We perform this block matching algorithm for each superblock/CTU, and shift each block towards bottom-right corner by half of the block size. Note that AIMC

**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
```



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

(b) Interpolated prediction for the entire partition structure.

Figure 3.4: Interpolated prediction built for the example partition in Fig. 3.2

is performed after motion vectors for all the required blocks have been selected. Therefore, instead of constructing off-grid blocks for an entire frame as in Chapter 2, we only construct off-grid blocks within each superblock/CTU, and ignore the blocks shifted outside the superblock/CTU boundaries. By confining the construction process within a superblock/CTU, the increase of the delay incurred by AIMC is more affordable.

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 an 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. 3.4 illustrates how interpolated predictions are done for the example partition in Fig. 3.2. Breaking the blocks using the proposed algorithm results in very limited off-grid block patterns, leading to reduction of complexity and simplification 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 [16].

There are three types of interpolated blocks (the blocks lie entirely within a superblock/CTU) in this structure. First, the blocks lying across the original block partition boundaries are valid blocks (solid blocks in Fig. 3.4), where AIMC can be performed. Second, the blocks lying entirely within the original blocks are non-valid interpolation blocks (dotted blocks in Fig. 3.4). 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. 3.4). 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.

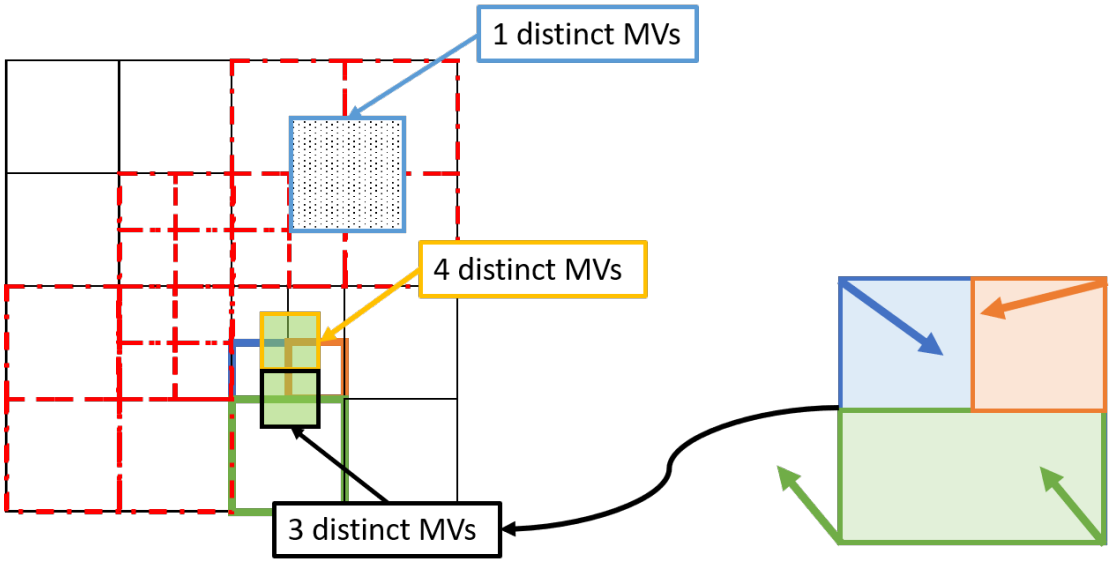



Figure 3.5: An example of the off-grid blocks in the same size have different number of distinct motion vector.

In addition to classify each off-grid block based on its size, we can further classify an off-grid block according to the number of distinct motion vectors in a block. Fig. 3.5 shows an example of the off-grid blocks in the same size having different number of distinct motion vectors. The black off-grid block has only up to three distinct motion vectors since the motion vector of the green standard block is assigned to the bottom corners of the black block. As how the interpolation coefficient can vary depends on the number of distinct motion vectors in the block and their constellation, further classify them according to this information can allow us to better account for each motion vector's influence. Note that there are a lot of possible constellations, but many of them are similar to each other up to rotation. For example, Fig. 3.6 shows all possible constellations with two distinct motion vectors. However, as many of them similar to each

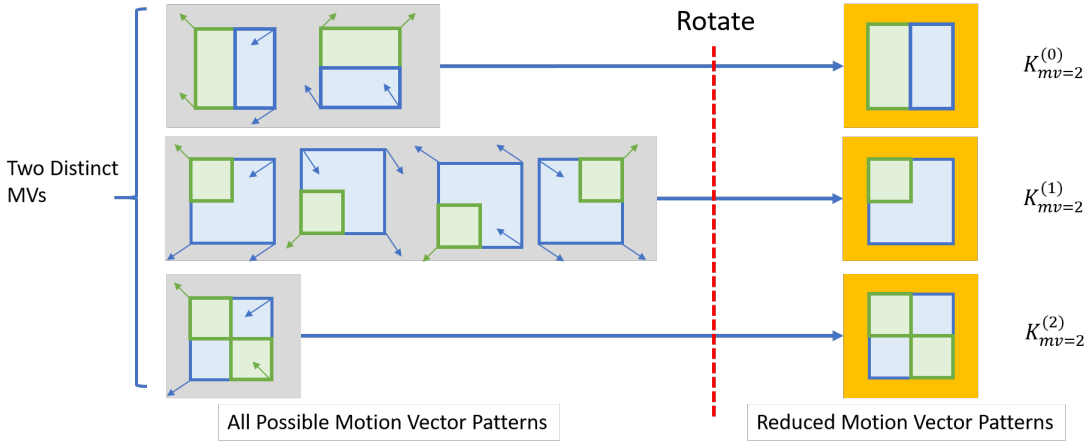



Figure 3.6: All possible motion vector constellations with two distinct motion vectors and their corresponding reduced motion vector pattern.

other with proper rotation, they can be grouped and trained together. When applying the coefficients, we just need to rotate the coefficients to match the actual constellation. Fig. 3.7 shows all possible unique motion vector patterns. Intuitively, the variations of the pattern with fewer distinct motion vector should be simpler. Therefore, we train fewer sets of interpolation coefficients for the simpler pattern to reduce the cost of side information.

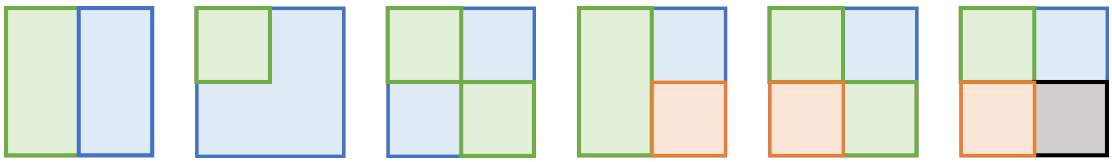


Figure 3.7: Possible motion vector patterns for the interpolated blocks (the sub-blocks in the same colors mean they share the same motion vector.).

Finally, we note that, in principle, we can form off-grid blocks to cover boundary area (horizontal-striped blocks in Fig 3.4b), and apply AIMC to those blocks. However, in practice, it is inefficient to perform AIMC on those off-grid blocks as

AIMC can only update the predictions to pixels within the current superblock/CTU due to the causality constraint. As a result, we skip those off-grid blocks that are partially outside the current superblock/CTU.

## 3.3 Deployment of Interpolated Prediction

Clearly, an area with complex texture and motions 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. 3.8 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.

(a) The reconstructed frame.

(b) Interpolated prediction area of the corresponding frame shown in Fig. 3.8a.

Figure 3.8: The reconstructed frame and the corresponding interpolated prediction area (colored with black).

## 3.4 Experimental Results

We evaluate the proposed non-parametric approach using the AV1 framework [2]. 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 $8 \times 8$, which limits the valid interpolated prediction block sizes to be $8 \times 8$, $16 \times 16$ and $32 \times 32$. We also do not allow compound predictions. 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 $K$ equals to the number of distinct motion vectors for each cluster shown in Fig. 3.7. We select $(\beta_x, \beta_y) \in \{(0,1),(1,0)\}$ for $K = 2$, $(\beta_x, \beta_y) \in \{(0,1),(1,0),(1,1)\}$ for $K = 3$, and $(\beta_x, \beta_y) \in \{(0,0),(0,1),(1,0),(1,1)\}$ for

$K = 4$. 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 at mid-target bit rate and then applied to all the different bit rate range. 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. 3.10. We also compare our results with using $K = 1$ for all the block sizes. We refer it as single mode where no side information is needed to indicate the selected set of coefficients used for interpolation. However, the encoder can still decide whether to enable this or not at superblock/CTU level based on RD cost. This can be viewed as generalization of the approach proposed in [14] on VBMC setting. The performance gain in terms of BD rate reduction is summarized in Table 3.1.

The intuition of choosing $K$ equals the number of distinct motion vectors in each motion vector is that we want to allow each motion vector can have a chance to be the dominant motion vector which can propagate its influence across its original boundary. Fig. 3.9 shows an example set of weight distribution, wherein

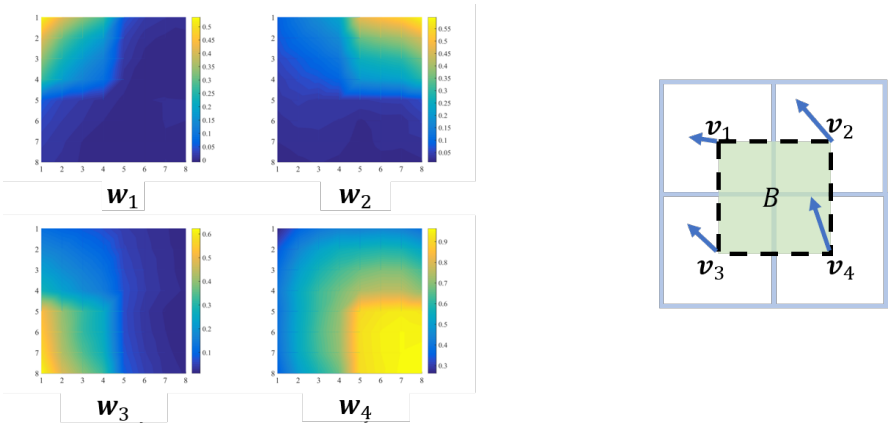



Figure 3.9: An example set of coefficients for the pattern with four distinct motion vectors and the corresponding motion vectors.

the bottom-right motion vector is the dominant motion vector.

From the results, we observe that for the sequences that are almost static or with mostly simple translation, such as *Container*, 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. Furthermore, compare with using single mode, our $K-$mode approach can achieve around 1% gain even an additional side information is required. This validates our claim that using single set of coefficients cannot well adapt to the local statistics especially for the

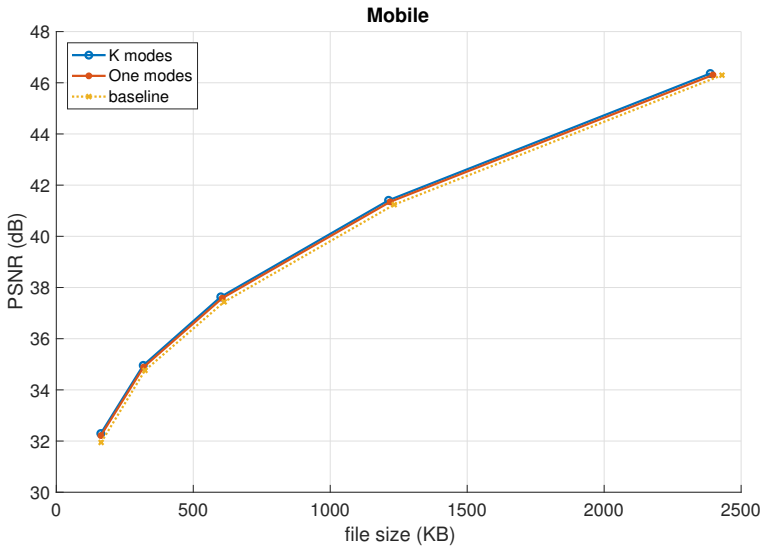



Figure 3.10: Coding performance comparison for sequence *Mobile*.

sequence with complex motion such as *Mobile* and *BQSquare*.

Table 3.1: BD rate reduction for the proposed approaches relative to AV1

| Sequence | Bit-Rate Reduction (%) (Single Mode) | Bit-Rate Reduction (%) (K-Mode) |
|---|---|---|
| Waterfall | -1.379 | -3.244 |
| Flower | -3.163 | -5.483 |
| Stefan | -1.872 | -2.601 |
| Mobile | -3.75 | -5.343 |
| Coastguard | -0.334 | -0.544 |
| Container | -0.586 | -0.804 |
| Ice | -0.299 | -0.351 |
| Foreman | -0.586 | -0.758 |
| Bus | -2.061 | -3.187 |
| BlowingBubbles | -1.791 | -2.607 |
| BQSquare | -3.393 | -5.757 |
| BQMall | -1.753 | -2.405 |
| Keiba | -0.155 | -0.313 |
| Average | -1.625 | -2.569 |

# Chapter 4

# AIMC - Parametric Approach

In the Chapter 2, we introduced a novel idea of incorporating neighboring motion vectors to generate more accurate prediction to a pixel. The method works very well in the fixed block size setting which produce a regular spacing motion grid. However, in the variable block size setting, the motion grid is irregular and there are different ways to define the off-grid blocks required in the method. As shown

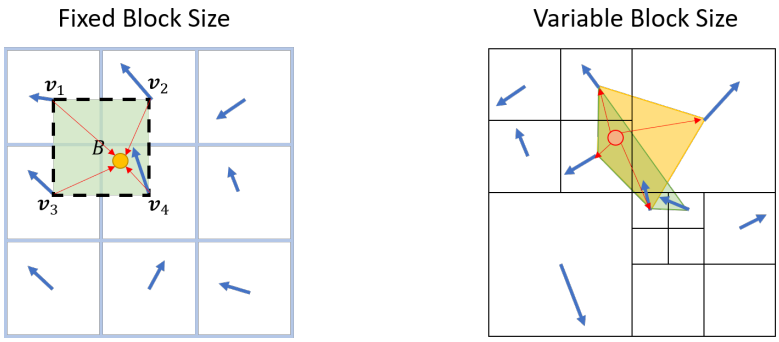



Figure 4.1: Motion grid generated by fixed block size and variable block size settings. In variable block size setting, the off-grid block definition in not unique. For example, there are two possible off-grid block patterns can contain the red pixel.

in the Fig. 4.1, the off-grid block that contains the red pixel is not unique. Furthermore, it is impractical to train and store coefficients for all possible off-grid block shapes. Therefore, in Chapter 3, we proposed an algorithm to limit the number of possible off-grid block shapes to effective train and store the coefficients. However, this approach also restricts the number of motion vectors we can use, and cannot operate on block boundary. It would be nice that given any number of relevant observations to a pixel at any location, we can derive the optimal coefficients to combine them to form the final prediction. To achieve this, we propose to use a parametric approach to model the second order statistics of the observations and the target. Specifically, we make use of the first-order Markov model to describe the correlations between pixels in a frame. The coefficients derived from our proposed model can automatically vary depending on both locations of motion vectors and their values to assign proper weights to each motion vector to adapt to local statistics. As result, the parametric can achieve better coding performance compared to the non-parametric (6.665% against 2.56% gains in term of bit-rate saving), as the parametric approach can operate on any location (therefore coverage area of interpolated prediction increased) and requires no additional side information to adapt to local statistics.

## 4.1 Motion Vector Selection at Encoder

We begin by introducing the motion vector selection process at a video encoder as this process indicates the prediction accuracy of using a neighboring motion vector as the predictor to the current block. Consider a simple block configuration as showed in Fig. 4.2, wherein two motion vectors $v_0$ and $v_1$ are assigned to the blocks through block-based motion compensation. When the difference between $v_0$ and $v_1$ are large, intuitively, it implies that these two blocks might belongs to different objects, and use one motion vector to predict the other block can also result in large error. Therefore, even the locations of the motion vectors are the same, the influences of the motion vectors to each other's block should vary depending on their values.

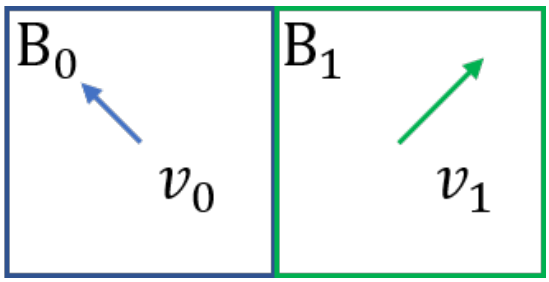



Figure 4.2: Block Configuration

In the block-based motion compensation, the motion vector of a standard block $B$ is selected to minimize the rate-distortion (RD) cost:

$$RD(\mathbf{v}) = \sum_{\mathbf{s}\in B}(x_k(\mathbf{s}) - \hat{x}_{k-1}(\mathbf{s}-\mathbf{v}))^2 + \lambda R(\mathbf{v}-\mathbf{v}_{\text{ref}}), \tag{4.1}$$

where $\mathbf{v}_{\text{ref}}$ is the reference motion vector, and $R(\cdot)$ is the rate required to encode the motion vector. The value of $R(\cdot)$ usually increases as the absolute value of its input increases. Therefore, the encoder uses one of its causal neighbors' (the blocks on the left or above the current block) motion vector as reference to reduce the value. In addition to serve as a reference, each of the causal neighbors' motion vectors is also used as the center point in the motion search process, and the prediction error of using it is computed. The encoder then selects the best motion vector and reference resulting minimum RD cost. Let $\mathbf{v}_0$ be a motion vector from a causal neighbor, and $\mathbf{v}^*$ be the optimal motion vector in terms of RD. We have the following relationships:

$$\begin{aligned}
&\sum_{\mathbf{s}\in B}(x_k(\mathbf{s}) - \hat{x}_{k-1}(\mathbf{s}-\mathbf{v}_0))^2 \geq \\
&\sum_{\mathbf{s}\in B}(x_k(\mathbf{s}) - \hat{x}_{k-1}(\mathbf{s}-\mathbf{v}^*))^2 + \lambda R(\mathbf{v}^*-\mathbf{v}_0) \geq \\
&\sum_{\mathbf{s}\in B}(x_k(\mathbf{s}) - \hat{x}_{k-1}(\mathbf{s}-\mathbf{v}^*))^2 + \lambda R(\mathbf{v}^*-\mathbf{v}_{\text{ref}}).
\end{aligned} \tag{4.2}$$

Without loss of generality we assume $R(0) = 0$. The lower bound of the prediction error using $\mathbf{v}_0$ increases as the difference between the selected motion vector $\mathbf{v}^*$ and $\mathbf{v}_0$ increases, implying that a neighboring motion vector is less reliable when the absolute difference between the motion vector is large. This provides a guideline of how the prediction from a neighboring motion vector should be weighted: the weights assigned to the prediction should decrease as the difference between the

two motion vectors increases. Even though the above property is derived from applying causal neighbors' motion vectors to the current block (use $v_0$ to predict $B_1$ as in Fig. 4.2), we assume the property is mutual, i.e., it also holds when non-causal neighbors' motion vectors are used to predict the current block (use $v_1$ to predict $B_0$ as in Fig. 4.2).

## 4.2 Optimal Linear Estimator

Given a list of motion vector candidates $\{\mathbf{v_0}, \mathbf{v_1}, \cdots, \mathbf{v_n}\}$, we can generate a vector of relevant observations, $[\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v_1}), \hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v_2}), \cdots, \hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v_n})]$, to a pixel locating at $\mathbf{s}$ on the frame $k$. The optimal linear coefficients that minimize the prediction error can be given by

$$\mathbf{c}(\mathbf{s}') = E[\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})^\intercal]^{-1} E[x_k(\mathbf{s})\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})]. \tag{4.3}$$

To derive the coefficients, we only need the second-order statistics: the auto-correlation of estimations, $E[\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})^\intercal]$, and the cross correlation between estimations and the target, $E[x_k(\mathbf{s})\hat{\mathbf{x}}_{k-1}(\mathbf{s}-\mathbf{v})]$. Therefore, if we can model these for any target pixel, we can compute the optimal coefficients to combine the observations.

### 4.2.1 Modelling the Auto-correlation Matrix

Each element in the auto-correlation matrix is the correlation between two observations which are essentially two pixels in the reference frame. The separable first order Markov model is known to be a good model to characterize the correlation between two pixels in nature images experimentally [17]. By imposing isotropic assumption, the correlation can be modeled as:

$$E\left[\hat{x}_{k-1}(\mathbf{s}-\mathbf{v}_{i,j})\hat{x}_{k-1}(\mathbf{s}-\mathbf{v}_{m,n})\right] = \rho^{||\mathbf{v}_{i,j}-\mathbf{v}_{m,n}||_1}, \tag{4.4}$$

where $\rho$ is the correlation coefficient. Note that when $\mathbf{v}_{i,j} = \mathbf{v}_{m,n}$, the estimates from both motion vectors are the same, and the correlation between them is naturally one. The correlation between two estimators $\mathbf{v}_{i,j}$ and $\mathbf{v}_{m,n}$ is characterized by the Manhattan distance $||\mathbf{v}_{i,j} - \mathbf{v}_{m,n}||_1$.

### 4.2.2 Modelling the Cross-correlation Vector

Similarly, the same first-order Markov model can be used to derive the correlation between an estimation and the target. Let the perfect estimation to the target pixel $\mathbf{s}$ locate at $\mathbf{s}^*$ which is pointed by $\mathbf{v}^*$ in the reference frame, i.e., we

assume $\hat{x}_{k-1}(\mathbf{s}^*) = x_k(\mathbf{s})$. The cross-correlation is:

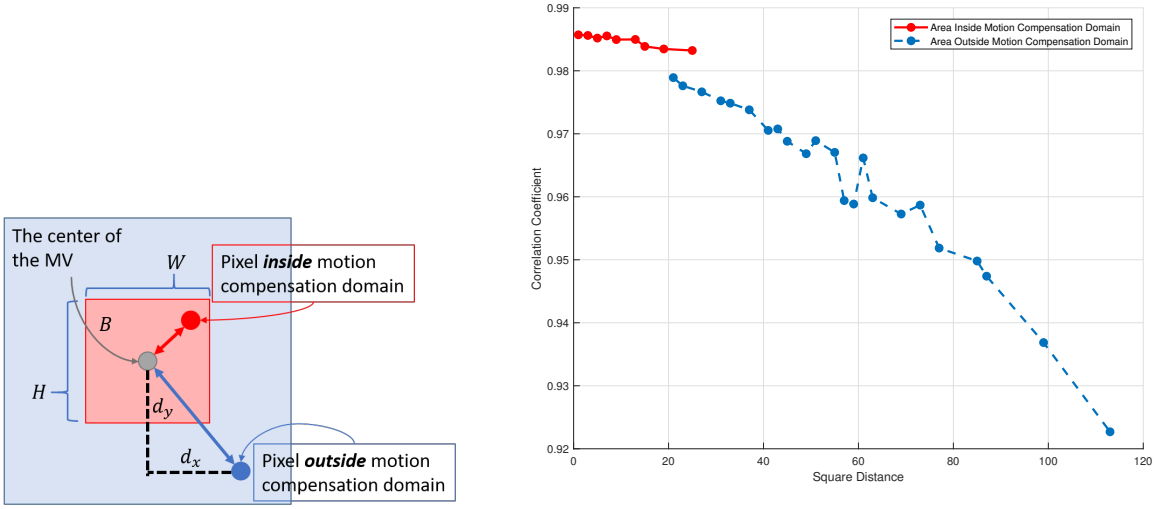
$$\begin{aligned} E\left[x_k(\mathbf{s})\hat{x}_{k-1}(\mathbf{s}-\mathbf{v}_{i,j})\right] &= E\left[\hat{x}_{k-1}(\mathbf{s}^*)\hat{x}_{k-1}(\mathbf{s}-\mathbf{v}_{i,j})\right] \\ &= \rho^{||\mathbf{s}-\mathbf{v}_{i,j}-\mathbf{s}+\mathbf{v}^*||_1} = \rho^{d(\mathbf{v}_{i,j},\mathbf{v}^*)}, \end{aligned} \tag{4.5}$$

where $d(\mathbf{v}_{i,j}, \mathbf{v}^*)$ is the Manhattan distance between an estimation and $\mathbf{s}^*$, which is unavailable to us. However, the observations are generated using motion vectors, and we can have the reliability of a motion to the target pixel and estimate the location of $\mathbf{s}^*$. Therefore, we will first describe how to model the reliability.

### Modeling Reliability of Motion Vectors

In general, the prediction error increases as the distance between a pixel and the motion vector (which is assumed to be located at the center of the block as shown in Fig. 4.3a) increases. Fig. 4.3b shows the correlation coefficients between target pixels and their predictions with respect to the distance between a pixel's location and the motion vector used to generate the prediction.

Note that there are two different areas. One is the area (the regular block $B$ defined in BMC) whose prediction error are considered in the selection of the motion vector. The other one is the area outside the motion compensation domain. There is a significant gap between these area since prediction accuracy outside the motion compensation domain is irrelevant to the selection of the motion vector and the error can be large. Therefore, we model the reliability of a motion vector




(a) Target pixel's location with respect to a motion vector.

(b) Correlations between the predictions and the targets.

Figure 4.3: Correlation Coefficients between the predictions and the target pixels inside/outside the motion compensation domain of a motion vector.

$\mathbf{v}$ to a pixel $\mathbf{s}$ as:

$$r(\mathbf{v},\mathbf{s}) = \begin{cases} \exp\left(-\alpha\left(\left(\frac{d_x(\mathbf{v},\mathbf{s})}{W}\right)^2 + \left(\frac{d_y(\mathbf{v},\mathbf{s})}{H}\right)^2\right)\right), & \mathbf{s} \in B \\ \beta \exp\left(-\alpha\left(\left(\frac{d_x^2(\mathbf{v},\mathbf{s})}{W}\right)^2 + \left(\frac{d_y^2(\mathbf{v},\mathbf{s})}{H}\right)^2\right)\right), & \mathbf{s} \notin B \end{cases} \tag{4.6}$$

where $\beta$ is a penalty for crossing block boundaries, $d_x(\mathbf{v},\mathbf{s})$ and $d_y(\mathbf{v},\mathbf{s})$ are the horizontal and vertical distances between the center of the motion vector $\mathbf{v}$ to the pixel $\mathbf{s}$ respectively; $W$ and $H$ are the size of the block $B$ on which conventional BMC is performed to generate the motion vector $\mathbf{v}$. We define the probability of

a motion vector $\mathbf{v}_i$ to be the true motion with probability:

$$p(\mathbf{v}_i) = \frac{r(\mathbf{v}_i, \mathbf{s})}{\sum_{\mathbf{v}_j \in \mathcal{N}} r(\mathbf{v}_j, \mathbf{s})} \tag{4.7}$$

**Modeling correlation between the target and observations**

Using the probability defined in eq. (4.7), we can compute the expected Manhattan distance between the target and an observation generated from a motion vector $\mathbf{v}_{i,j}$:

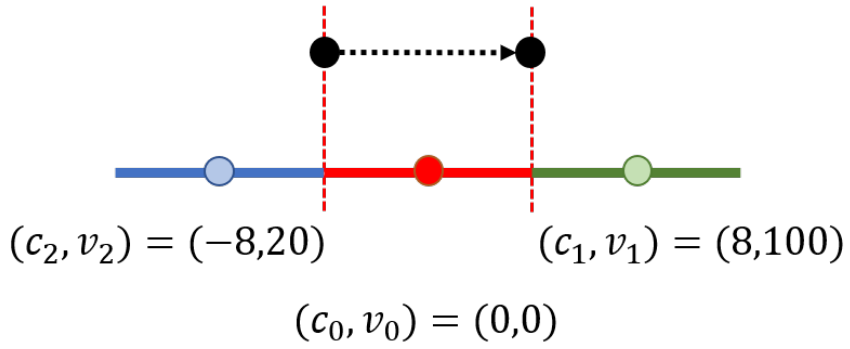
$$E\left[d(\mathbf{v}_{i,j}, \mathbf{v}^*)\right] = \sum_{\mathbf{v}_{m,n} \in \mathcal{N}} p_{m,n} ||\mathbf{v}_{i,j} - \mathbf{v}_{m,n}||_1, \tag{4.8}$$

where $\mathcal{N}$ is the set containing all the motion vectors we used to generate estimations to the target pixel. We model the correlation between the target and the observation as:

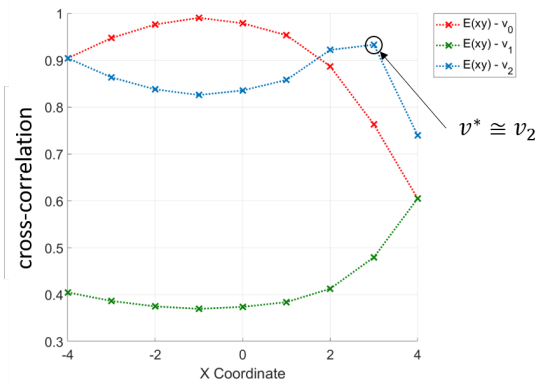
$$E\left[x_k(\mathbf{s})\hat{x}_{k-1}(\mathbf{s} - \mathbf{v}_{i,j})\right] = \rho^{E[d(\mathbf{v}_{i,j}, \mathbf{v}^*)]} \tag{4.9}$$

By using this approach, we can circumvent computing the distance through guessing some intermediate proxy $\hat{\mathbf{s}}^*$ to prevent from some undesired results. To illustrate this, let us consider a simple 1-D example as shown in Fig. 4.4a wherein the three motion vectors locate at $-8, 0$ and 8 respectively. With the probabilities, one tempting approach would be naively defined $\hat{v}^* = E(\mathbf{v})$ and $\hat{\mathbf{s}}^* = E(v) + \mathbf{s}$.

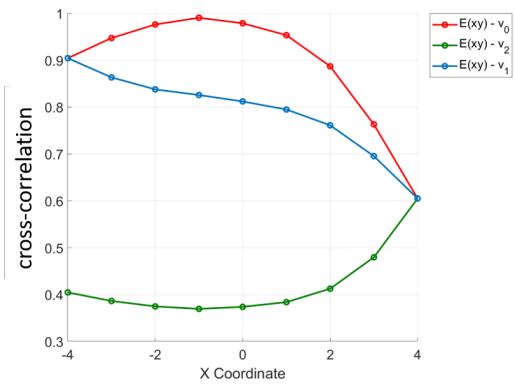
However, using this approach, the expected motion around the boundary between the red and the green block will be very close to $v_2$, resulting the estimated cross-correlation increase as it moves further away from the location of $v_2$ as shown in Fig. 4.4b. This trend does not match to the actual statistics. It is only valid if the underlying motion changes smoothly between the motion vectors. On the other hand, using the proposed method, the derived cross-correlation changes more properly on the boundary area.




(a) Motion vectors configuration: the motion vectors $v_0 = 20, v_1 = 0, v_2 = 100$ locate at $-8, 0, 8$ respectively.




(b) Naive approach: $\hat{v}^* = E(v)$ and $\hat{\mathbf{s}}^* = E(v) + \mathbf{s}$




(c) Proposed approach

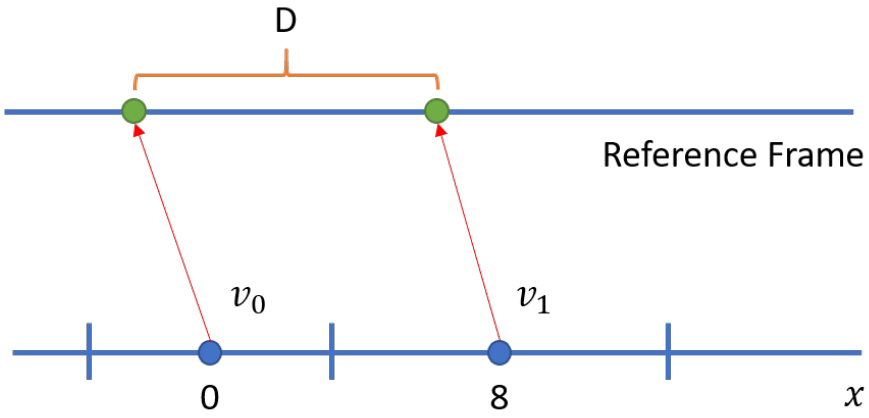
Figure 4.4: Example of motion vectors configuration and the estimated cross–correlation for the center (red) block.

### 4.2.3 Overall scheme

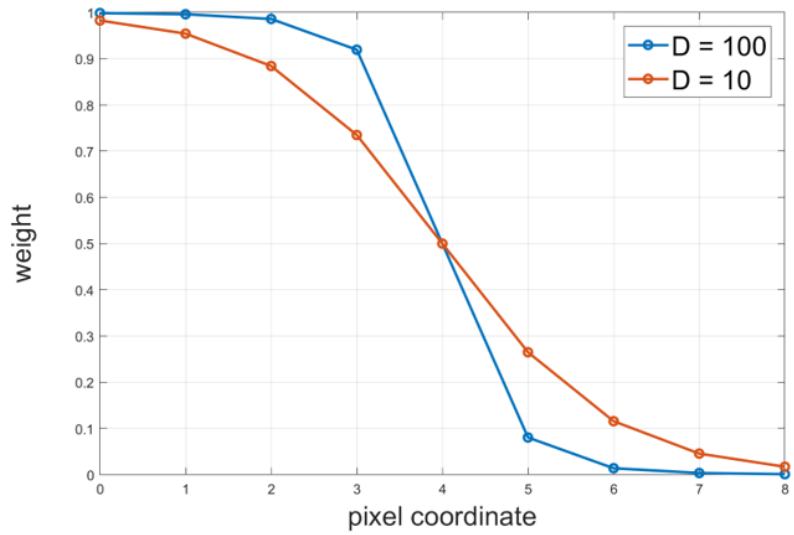
Here we summarize the overall scheme of our proposed method:

1. For each pixel, collect the motion vector candidates from neighboring blocks.
2. Estimate auto-correlation matrix using eq. (4.4).
3. For each pixel, derive the probability assigned to each motion vector using (4.6) and estimate the cross-correlation vector using eq. (4.9)
4. Calculate the coefficient using eq. (4.3).

Note that step 1 and 2 can be done per block. Therefore, the auto-correlation matrix is the same for all the pixel in the same block. We can then compute and store the inverse of the auto-correlation matrix, and therefore derivation of the coefficients can be very fast. Fig. 4.5 shows an example of coefficient distribution of using two motion vector candidates. The coefficients are different even though the motion vectors' locations are fixed, and they decrease sharply when the difference between the two motion vectors, $D$, grows large. As discussed in subsection 4.1, the ways the coefficients vary agree with the prediction accuracy of the motion vector, and therefore the proposed model can adapt to local statistics.




(a) Example of motion vector configure: $v_0$ locates at 0 and $v_1$ locates at 8, and $D$ is the distance between prediction in the reference frame.




(b) The weights assigned to prediction from $v_0$.

Figure 4.5: An illustration of weight distribution of parametric interpolation (a two motion vectors case).

## 4.3 Generalization for Multi-reference Frames

Using eq. (4.4) and (4.9) we can derive the cross and auto correlations when the motion vectors point to the same reference frame. However, in current video coders, multiple previously reconstructed frames can be served as reference frames

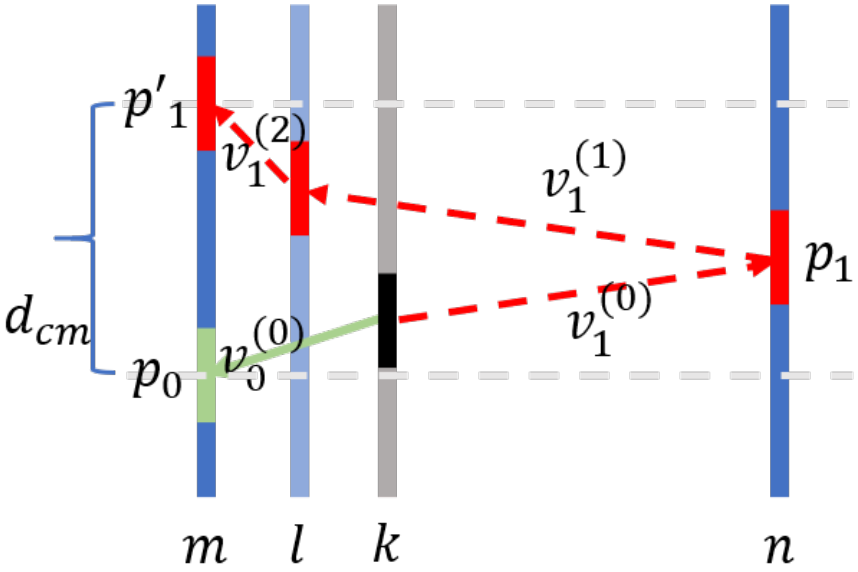



Figure 4.6: When the predictions are in different reference frames, the distance in first common frame along the prediction chain is used to compute the correlation.

to predict the current frame. The questions raised on how to compute the correlations between the observations when the motion vectors points to different reference frames. Note that to derive the correlations, we only need to find the distance between two observations or their motion vectors, yet as the scenes or objects in a video sequences are barely static, the direct spatial distance (assuming pixels are not moving) is not meaningful. To solve this, we propose the following two methods to account for a pixel's trajectory to find the correct distance for our model.

### Prediction Chain Tracking

Consider the example showed in Fig. 4.6 where the two motion vectors $v_0^{(0)}$ and $v_1^{(0)}$ generates two estimates $p_0$ and $p_1$ respectively. The estimate $p_1$ is predicted

through the motion vector $v_1^{(1)}$ which points to a pixel predicted through another motion vector $v_1^{(2)}$. We refer this prediction structure as a prediction chain. Even though $p_1$ and $p_0$ are on different reference frames, as we track along the prediction chain of $p_1$, we can find $p_1'$ which is on same reference frame as $p_0$. The prediction $p_1'$ is very similar to $p_1$ as the encoder use pixel domain block matching method for motion compensation. Therefore, the correlation of $p_1'$ and $p_0$ can be used to approximate the correlation of $p_1$ and $p_0$, and we define the distance between the motion vectors $\mathbf{v}_0^{(0)}, \mathbf{v}_1^{(0)}$ as:

$$d_{cm}(\mathbf{v}_0^{(0)}, \mathbf{v}_1^{(0)}) = \left|\left| \sum_{i=0}^{m^*} \mathbf{v}_0^{(i)} - \sum_{j=0}^{n^*} \mathbf{v}_1^{(j)} \right|\right|_1, \tag{4.10}$$

where

$$m^*, n^* = \arg\min_{m,n} m + n \tag{4.11}$$

$$\text{such that } v_0^{(m^*)} \; v_1^{(n^*)} \text{ point to the same reference frame.} \tag{4.12}$$

The correlation of the two estimations can be computed as:

$$E\left[\hat{x}_m(\mathbf{s} - \mathbf{v}_0)\hat{x}_n(\mathbf{s} - \mathbf{v}_1)\right] = \rho^{d_{cm}(\mathbf{v}_0, \mathbf{v}_1)}$$

We note that if $p_1$ is predicted using AIMC, then there will be multiple motion vectors contribute to the prediction of $p_1$. In that case, we only use the main

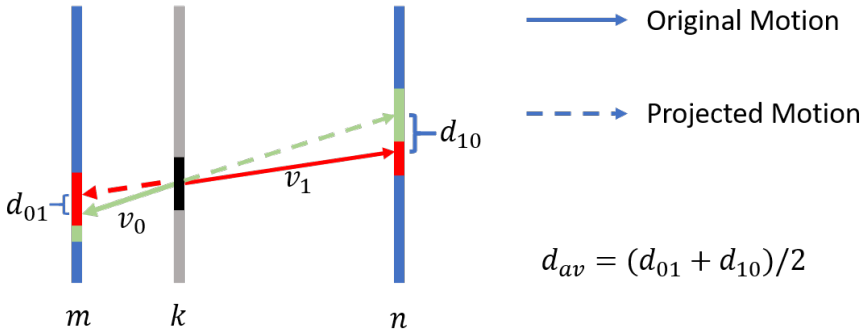



Figure 4.7: When the motion vectors point to different reference frames, and the prediction chain method is not applicable. Linear motion is assumed, and the motion vectors are extended to the each other's reference frame. The average distance between the original motions and projected motions are used to define the distance between the two original motion vectors $v_0$ and $v_1$.

motion vector of $p_1$ (the motion vector assigned by VBMC) to account for the correlation between the predictions $p_0$ and $p_0$ for simplicity. We also note that since the prediction residuals are in general non-zero, the approximation to the correlation will be more and more inaccurate as we track deeper along the prediction chain. Ideally, this can be solved by comparing predictions $p_1$ and $p_1'$ to estimate the temporal correlation between the motion trajectory and incorporate it in our model. However, it requires to generate additional predictions and the benefit is not significant. As a result, we simplify the model by assuming the temporal correlation is one, and use this method only when the prediction of one of the two estimates and the other estimate are on the same reference frame, i.e. $m^*+n^* = 1$. Similarly, the cross-correlation is computed by replacing $||\mathbf{v}_{i,j} - \mathbf{v}_{m,n}||_1$ in eq. (4.8) with $d_{cm}(\mathbf{v}_{i,j}, \mathbf{v}_{m,n})$.

**Motion Vector Extension**

The prediction chain described in the previous method can also end if a pixel is predicted using intra-prediction. Therefore, other than the scenario wherein the first common reference frame is too far away along the prediction chains, the first method is simply not applicable when a prediction chain ends before we can find a common reference frame. When either one of these two situations happens, we assume pixels move linearly and extend the motion vectors to each other's reference frames to derive the distance for computing correlations between the estimates. Specifically, let $v_0$ and $v_1$ point to frame $m$ and $n$ respectively as shown in Fig. 4.7. We extend the motion vector to each other's frame and compute:

$$d_{ext}(\mathbf{v}_0, \mathbf{v}_1) = \left|\left|\mathbf{v}_0 - \frac{m-k}{n-k}\mathbf{v}_1\right|\right|_1 \text{ and } d_{ext}(\mathbf{v}_1, \mathbf{v}_0) = \left|\left|\mathbf{v}_1 - \frac{n-k}{m-k}\mathbf{v}_0\right|\right|_1, \quad (4.13)$$

where $k$ is the current frame. We define the distance between the two motion vectors to be the average value of the distances on the two frames:

$$d_{av}(\mathbf{v}_0, \mathbf{v}_1) = \frac{d_{ext}(\mathbf{v}_0, \mathbf{v}_1) + d_{ext}(\mathbf{v}_1, \mathbf{v}_0)}{2} \quad (4.14)$$

Then the correlation between the two estimates is

$$E\left[\hat{x}_m(\mathbf{s} - \mathbf{v}_0)\hat{x}_n(\mathbf{s} - \mathbf{v}_1)\right] = \rho^{d_{av}(\mathbf{v}_0, \mathbf{v}_1)} \quad (4.15)$$

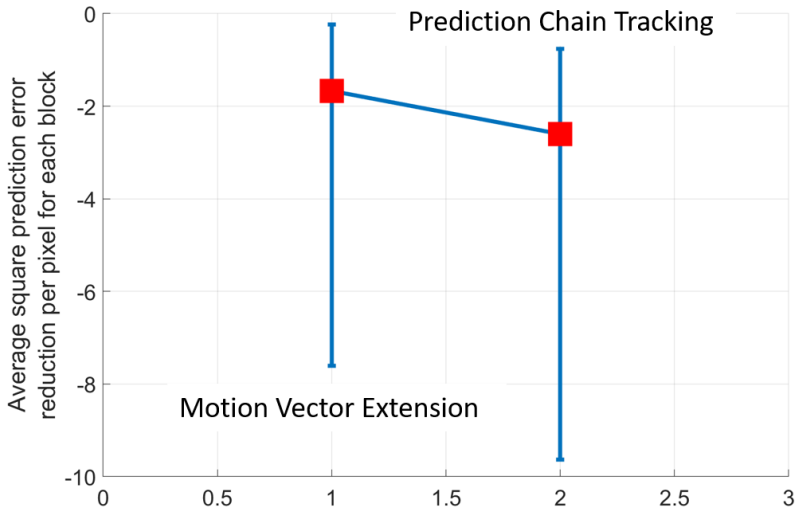



Figure 4.8: The distributions of average square prediction error reduction of using two different strategies to model the correlation of estimates from two motion vectors pointing to different reference frame. The red dots denote the median, and the error-bars shows the first to third quantiles.

### Comparison between the two proposed methods

The *prediction chain tracking* method utilizes the motion information to account for the trajectory of a pixel and therefore can describe their correlation more accurately, while the *motion vector extension* method is more general and can be applied to any scenario. Fig. 4.8 shows the distributions of averaged squared prediction error reduction of a block using both methods when they are all applicable and motion vectors point to different reference frames. It shows that indeed, using the *prediction chain tracking* method we can get more accurate prediction. Therefore, we try to compute the distance between the two motion vectors according to the following order:

1. Manhattan distance if they point to the same reference frame.

2. Prediction chain tracking if the first common reference frame can be reached in one step (eq. (4.11) = 1).

3. Motion vector extension.

## 4.4 Experimental Results

We now describe the experiments and the results of using the proposed parametric approach. Since the focus of this work is on breaking the block structure and optimally weighting the influence of each motion vector in the motion field. We first test our method using the encoder configuration described in Chapter 3 wherein the compound (bi-directional) prediction mode is disabled, as when compound mode is used, the encoder specifically searches for two motion vectors to create the final prediction to a block. This compound prediction mode can be viewed as a block-based version of multi-hypothesis prediction using two estimators. Therefore, the functionality of proposed approach overlaps with the compound prediction mode to some extent. Then we enable the compound mode and test our method on the baseline without the restriction. The correlation coefficients between pixels $\rho$ in eq. (4.4) is set to be 0.99, which is obtained by analyzing the pixel correlation within the prediction block of *Keiba* sequence, and for the parameters in eq. (4.6), we set $\alpha = 0.025$, $B = 0.8$.

Table 4.2 summarizes the experimental results. The third column shows the

BD-rate saving over the baseline with the compound mode disabled. Compared with the non-parametric AIMC, the gain from parametric method is doubled. There are several factors involved. First, the parametric approach uses the "actual" location (center of a block) of a motion vector to derive the its influence onto its neighbor, while in the non-parametric approach, we divide a large block into smaller ones and therefore the location of the motion vector is shifted. Second, the boundary area of a superblock is skipped due to inefficiency to spend additional side information on those off-grid blocks. For a superblock/CTU partitioned into $16 \times 16$ blocks, the boundary area comprises of around 46% of entire superblock/CTU. However, the parametric approach can be applied on any location, and therefore the performance increases as coverage area of AIMC increases. Finally, the coefficients derived from the parametric model can automatically adapt to local statistics. The additional side information to indicate which sets of coefficients in the non-parametric approach is saved.

The fourth column of Table 4.2 shows the BD-rate saving over the baseline where the compound mode is enabled. We can observe that the gain reduces largely when the compound mode is enabled, wherein the encoder is allowed to use two different motion vectors for a blocks, and combine the predictions from the motion vectors with proper weights. The compound mode is used when a single motion vector cannot produce a satisfactory prediction. As we can observe from the table, for the video sequences has complex textures or complicated motions

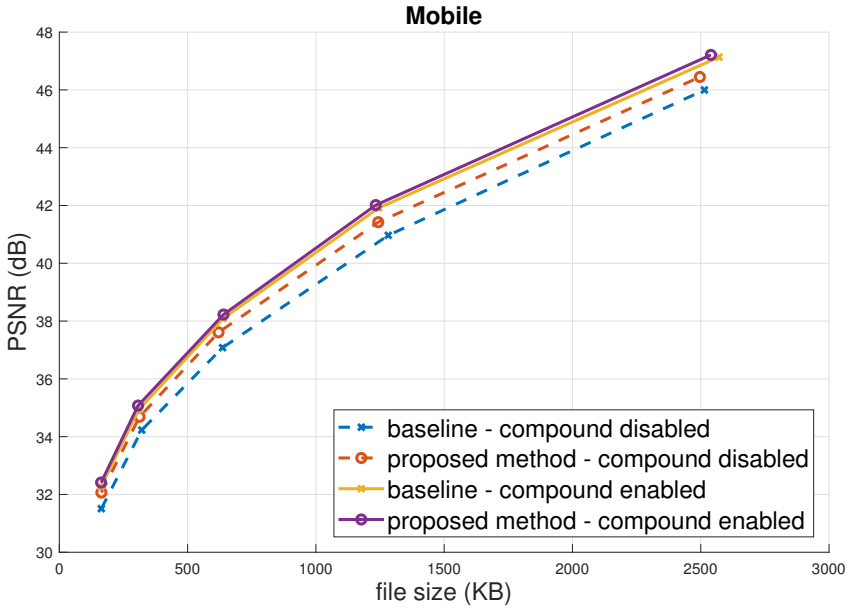
(such as *Waterfall*), the encoder tend to use the compound mode to generate more accurate predictions. The percentage of inter-predicted area predicted by the compound mode ranging from $60\% - 70\%$ for that kind of videos. Fig. 4.9 shows that, with compound mode enabled, the prediction is more accurate, and the coding performance increases correspondingly. Note that the encoder specifically searches for a second motion vector to minimize RD cost when the compound mode is used. As shown in Table 4.1, this increases computational complexity at the encoder side by more than 130%, while our AIMC simply reuses the neighboring motion vectors and weights their influences in a proper way. The AIMC approach only increases the complexity by about 21% but is able to retain most of the gain achieved by the compound prediction mode. We observe that one of the two motion vectors of a compound-predicted block is usually the same or very close to one of its neighboring motion vectors. In this scenario, the inaccurate prediction of using a single motion vector can be addressed by our AIMC approach later. We also note that, as shown in Fig. 4.10, the encoder picks the compound mode more often when encoding the high-resolution videos in our test set. Therefore, AIMC has larger gain for low resolution than for high resolution video in Table 4.2.

One way to further boost the performance of AIMC is to incorporate it into the motion vector selection loop of a block. In our current implementation, the encoder is not aware of the AIMC when selecting motion vectors, and therefore

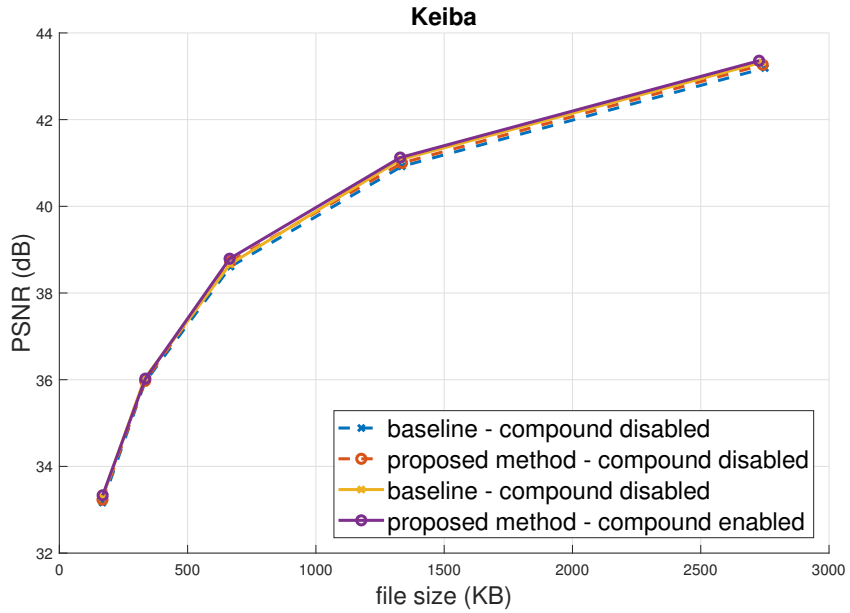
Table 4.1: Computational complexity increments against baseline (with the compound mode disabled) at the encoder.

| AIMC (with compound mode disabled) | Compound Mode Enabled |
|---|---|
| +21.14% | +139.01% |

will choose to use the compound mode even though AIMC can provide a good prediction afterward. By incorporating AIMC into the loop, the encoder can properly weight the impact of neighboring motion vectors and form a more accurate prediction without using a second motion vector (which needs additional side-information) at the very beginning.




(a) $M$obile

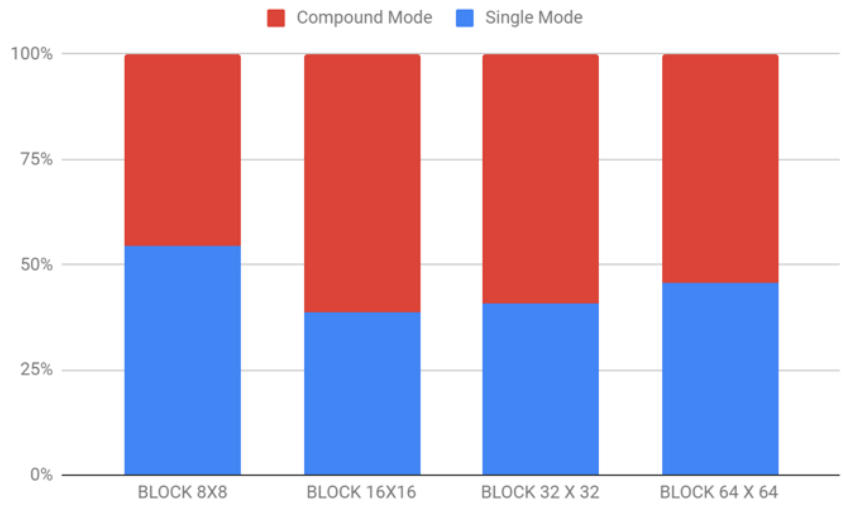



(b) $K$eiba

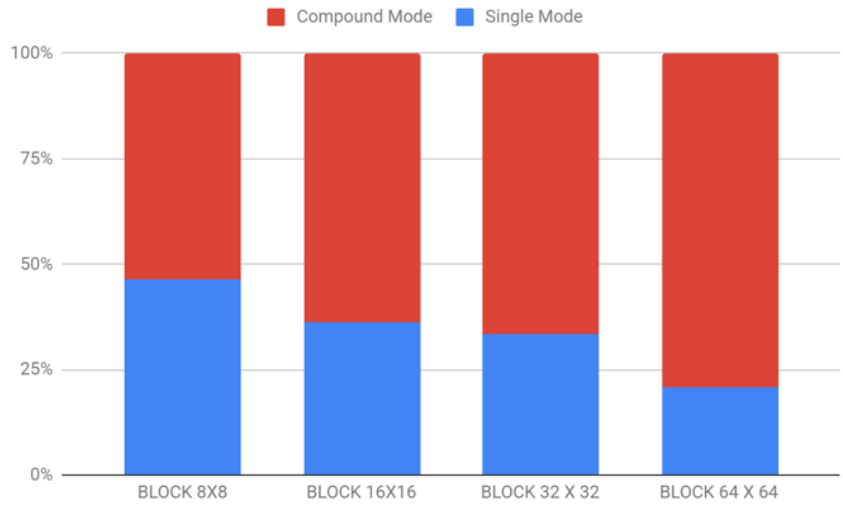
Figure 4.9: Coding performance comparison.

Table 4.2: BD rate reduction for the proposed approaches relative to AV1. The last column shows the percentage of inter-predicted area used compound mode when the mode is enabled.

| Resolution | Sequence | Bit-Rate Reduction (%) (Compound Mode Disabled) | Bit-Rate Reduction (%) (Compound Mode Enabled) | % of Compound Mode Used |
|---|---|---|---|---|
| CIF | Waterfall | -11.358 | -2.871 | 78.9 |
| | Stefan | -6.008 | -1.982 | 55.7 |
| | Mobile | -10.902 | -2.72 | 73.0 |
| | Container | -5.231 | -0.139 | 79.4 |
| | Ice | -2.131 | -0.949 | 28.5 |
| | Bus | -6.135 | -2.082 | 60.7 |
| | Flower | -8.728 | -2.314 | 59.2 |
| | Coastguard | -5.078 | -0.64 | 74.5 |
| | Foreman | -5.240 | -2.161 | 44.5 |
| 416 × 240 | BlowingBubbles | -7.318 | -1.672 | 68.6 |
| | BQSquare | -12.565 | -1.728 | 78.0 |
| 832 × 480 | Keiba | -2.071 | -1.329 | 27.2 |
| | BQMall | -3.884 | -0.594 | 55.4 |
| 1080 × 720 | Shields | -7.972 | -0.822 | 78.0 |
| 1920 × 1080 | BasketballDrive | -3.427 | -0.714 | 67.0 |
| | BQTerrace | -10.037 | -0.935 | 71.3 |
| | Cactus | -7.076 | -0.998 | 69.7 |
| | Kimono1 | -3.416 | -0.672 | 68.6 |
| | Rush hour | -2.07 | -0.616 | 48.9 |
| | Average | -6.665 | -1.629 | |




(a) Statistics for low resolution video sequences ($352 \times 240$ and $416 \times 240$).




(b) Statistics for high resolution video sequences ($1080 \times 720$ and $1920 \times 1080$).

Figure 4.10: Average percentage of inter-prediction modes usage for four different block sizes.

# Chapter 5

# Advances in Hierarchically Structured Multi-Reference Prediction

In this Chapter, we introduce a multi-layer multi-reference prediction framework and a new compound prediction mode for effective video compression. Current AOM/AV1 baseline uses three reference frames for the inter prediction of each video frame. This Chapter first presents a new coding tool that extends the total number of reference frames in both forward and backward prediction directions. A multi-layer framework is then described, which suggests the encoder design and places different reference frames within one Golden Frame (GF) group to different layers. The multi-layer framework leverages the existing coding tools in the

AV1 baseline, including the tool of "show_existing_frame" and the reference frame buffer update module of a wide flexibility. The use of extended ALTREF_FRAMEs is proposed, and multiple ALTREF_FRAME candidates are selected and widely spaced within one GF group. ALTREF_FRAME is a constructed, no-show reference obtained through temporal filtering of a look-ahead frame. In the multi-layer structure, one reference frame may serve different roles for the encoding of different frames through the virtual index manipulation. Finally, as reference frames cover a wider range, the original compound prediction modes are insufficient to utilize the temporal variations. We introduce a new compound prediction mode, selectively weighted compound mode, which allows the encoder to weight the predictions from different reference frames more flexible. The experimental results have been collected over several video test sets of various resolutions and characteristics both texture- and motion-wise, which demonstrate that the proposed multi-layer approach achieves a consistent coding gain compared to the AV1 baseline. For instance, using PSNR as the distortion metric, an average bitrate saving of 5.57+% in BDRate is obtained for the CIF-level resolution set, some of which has a gain of up to 13+%, and 4.47% on average for the VGA-level resolution set, some of which up to 18+%. In addition, with the new compound prediction mode, we can further boost the performance by in average 0.6%.

## 5.1 Introduction

Google embarked on the open-source project entitled WebM [18] in 2010 to develop open-source, royalty unencumbered video codecs for the Web. WebM released two editions, first VP8 [19] and then VP9 [6], where VP9 achieves a coding efficiency similar to the latest video codec from MPEG entitled HEVC [4]. VP9 has delivered a huge improvement to YouTube in terms of quality of experience metrics over the primary format H.264/AVC [20]. Google then joined the Alliance for Open Media (AOM) [2] effort for a Joint Development Foundation project formed with a few other industrial leaders, to define and develop media codecs, media formats, and related technologies, still under the open standard. In this Chapter, we focus on the multiple reference inter prediction aspect for the to-be first edition of the AOM video codec, namely AV1.

The use of multiple reference frames facilitates a better inter prediction for videos with a variety of motion characteristics, such as the presence of occlusion and uncovered objects, lighting changes, fade-in and fade-out effects, static background, etc. The state-of-the-art techniques proposed the use of both short-term references and long-term references (LTR) [21] to adapt to the specific content and motion features presented in the coded frame. The Rate-Distortion (RD) performance optimization requests a trade-off between identifying the best reference for one coded frame and the overhead bits spent in signaling the multi-reference candidates [22–24]. Further, the encoder-side computational complexity should be

considered [25]. Leveraging the multiple reference resources, one video frame may be forward predicted or backward predicted or both, referred to as bidirectionally predicted [26]. Special modes have been designed to effectively encode these bi-predictive frames, i.e. B frames, including the use of DIRECT mode [27, 28] and the design of hierarchical B frames [29].

In this Chapter, we first introduce a new coding tool that extends the number of reference frames in AV1 from three to six to increase the flexibility and adaptability for the multi-reference prediction. Furthermore, we describe the encoder design through the exploit of extended `ALTREF_FRAME`s, and form a multi-layer framework facilitated by the two coding tools provided in AV1, namely the "show_existing_frame" and the virtual index manipulation. Finally, we introduce a new compound prediction mode, selectively weighted compound mode, which allows the encoder to weight the prediction from different reference frames more flexible to best utilize the wide temporal variations from more diverse references provided by the new structure. The experimental results validate the efficiency of the multi-layer structure and the new compound mode with a consistent coding gain compared to the AV1 baseline over a variety of video test sets in various resolutions.

## 5.2 A New Coding Tool

### 5.2.1 AV1 Baseline Reference Frame Design

Current AOM/AV1 baseline uses three reference frames for the coding of each inter-coded frame: `LAST_FRAME`, `GOLDEN_FRAME`, and `ALTREF_FRAME`. The three references used by one specific coded frame are selected from a reference frame buffer that can store up to eight frames. In general, an AV1 encoder may select `LAST_FRAME` from a near past frame, and `GOLDEN_FRAME` from a distant past. `ALTREF_FRAME` is a no-show frame usually constructed from a distant future frame through temporal filtering. An AV1 encoder may apply different temporal filtering strength to construct an `ALTREF_FRAME`, adapting to various motion smoothness levels across frames. A so-called Golden Frame (GF) group can be established, and all the frames within one GF group may share the same `GOLDEN_FRAME` and the same `ALTREF_FRAME`. `LAST_FRAME` may be updated constantly. When the distant future frame that provides `ALTREF_FRAME` is actually being coded, it is referred to as an `OVERLAY` frame but treated as a regular inter frame. `OVERLAY` frames usually cost fairly small amounts of bits as `ALTREF_FRAME` may serve as an ideal prediction.

AV1 baseline designs two types of inter prediction: A block predicted from one reference frame with a corresponding motion vector is said to be in a *single prediction* mode, while a block predicted using two different reference frames and

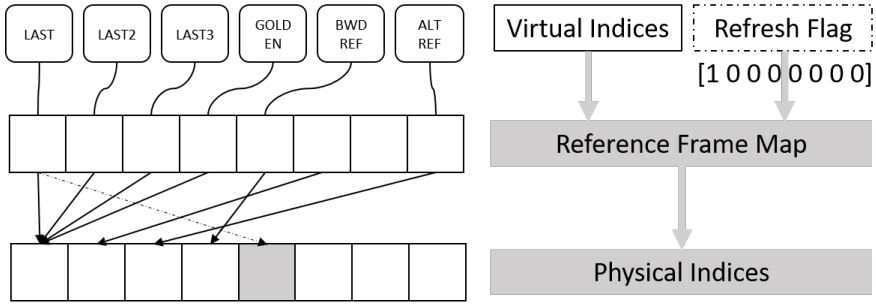



Figure 5.1: AV1 reference frame buffer update.

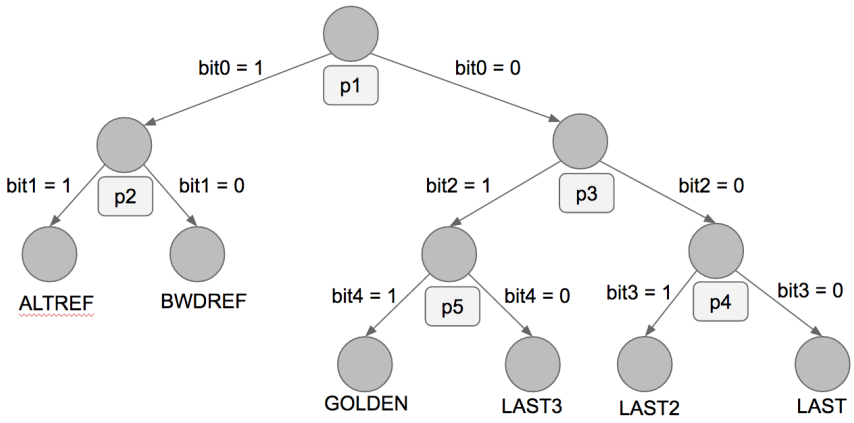
two corresponding motion vectors is said to be in a *compound* mode. *Compound* prediction always chooses the two predictions from two different directions, and generates a new predictor by simply averaging the two *single* predictors.

The reference frame buffer update in AV1 is realized through two syntaxes in the frame level: First is an eight-bit reference Refresh Flag, with each bit signaling whether the corresponding frame in the reference buffer needs to be refreshed or not by the newly coded frame; The second syntax is a mechanism referred to as "Virtual Index Mapping", as shown in Fig. 5.1. Each of the three references is labeled by a unique virtual index, and both the encoder and the decoder maintain a Reference Frame Map to associate a virtual index with the corresponding physical index that points to its location within the reference buffer. Both the Refresh Flag and the virtual indices are written into the bitstream. The advantage of using such mapping mechanism is to largely avoid memory copying whenever reference frames are being updated.

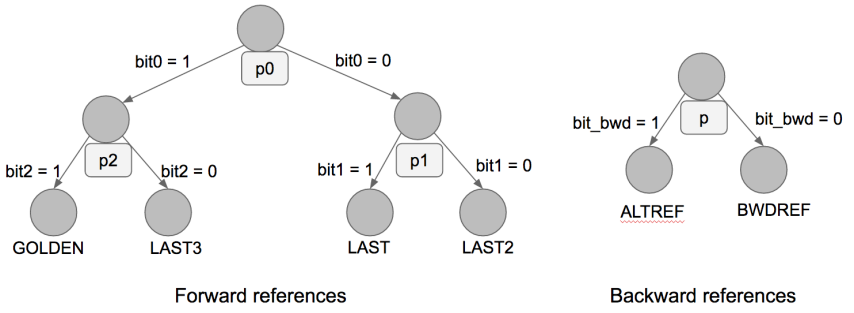
### 5.2.2 Extended Reference Frame - A New Coding Tool

To make full use of the reference frame buffer designed to store a maximum of eight frames, we propose a new coding tool that extends the number of reference frames for each coded frame from three to six. Specifically, we add `LAST2_FRAME`, `LAST3_FRAME`, and `BWDREF_FRAME`, where the former two references are usually selected from past for forward prediction and the later selected through look-ahead for backward prediction. Moreover, different from `ALTREF_FRAME`, `BWDREF_FRAME` leverages the existing coding tool provided by the AV1 baseline, namely the "show_existing_frame" feature, to encode a look-ahead frame without applying temporal filtering, thus no corresponding `OVERLAY` frame is needed. The use of `BWDREF_FRAME` is more applicable as a backward reference at a relatively shorter future distance. The extended reference frames allow a total of six candidates for the *single prediction* mode, and a total of 8 candidates for the `compound` mode as a combination of a forward predictor and a backward predictor are considered. Consequently, each video frame is offered an extensively larger set of multi-reference prediction modes, thus leading to a great potential for the rate-distortion (RD) performance improvement.

To efficiently encode the extended number of references, context-based, bit-level binary tree structures are adopted, as shown in Fig. 5.2a and Fig. 5.2b. Depending on the availability and the final coding modes of the two neighboring blocks within the causal window - on the top and at the left, five contexts




(a) *single reference* prediction




(b) *compound* prediction

Figure 5.2: Binary tree structure design for context-based, bit-level entropy coding of the extended reference frames.

are designed for the coding of every bit in either *single reference* or *compound* prediction.

Moreover, through the use of `BWDREF_FRAME`, a symmetric framework of multi-reference prediction is established for the *compound* mode: (1) A `BWDREF_FRAME` may be selected from a nearer future frame, paired with the nearer past `LAST_FRAME`; (2) A `BWDREF_FRAME` may be selected from a father future frame, paired with the father past `LAST2_FRAME`; and (3) `ALTREF_FRAME` may be selected from a distant future frame, paired with the `GOLDEN_FRAME` in the distant past. The use

of extended reference frames that are spread out widely thus allows an adaptation to the dynamic motion characteristics within one video sequence.

## 5.3 Encoder Design - A Multi-Layer Framework

Aligned with the new coding tool introduced in Session 5.2, we address the encoder design in this session. An extended ALTREF_FRAME scheme is proposed, which adopts more than one ALTREF_FRAME candidates within one GF group. Still complied with the syntax that allows one ALTREF_FRAME at maximum for the coding of each frame, several frames may be buffered to act as ALTREF_FRAME serving for different frames. These candidates may be selected from various locations within the GF group and have various temporal filtering strengths applied. A multi-layer framework is then constructed with the aid of the extended ALTREF_FRAMEs. Such encoder design is targeted to make full use of the eight-frame spots in the reference buffer and best leverage the new coding tool of extended reference frames.

### 5.3.1 Extended ALTREF_FRAMEs

As illustrated in Fig. 5.1, the "Virtual Index Mapping" mechanism specifies how the reference frame buffer is updated. Both the encoder and the decoder use identical virtual indices associate with the same reference frame, and maintain a respective Reference Frame Map to track the corresponding physical location in

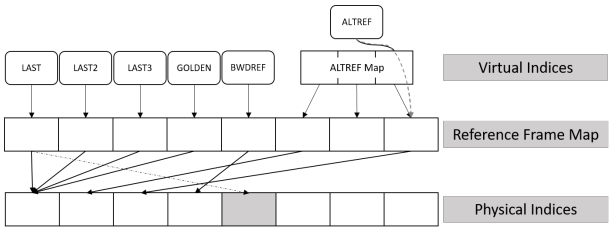
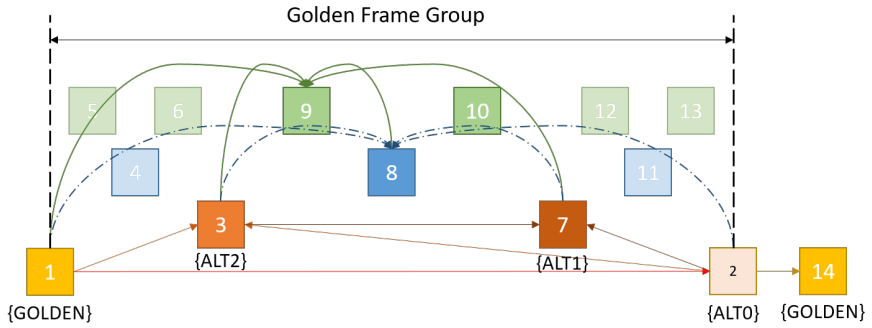



Figure 5.3: Encoder design using extended ALTREF_FRAMEs.

the reference frame buffer. Within one GF group the encoder may buffer multiple frames to serve as the ALTREF_FRAME candidates, which is referred to as the extended ALTREF_FRAME scheme. To facilitate such an encoder design, an ALTREF Map is exploited only at the encoder side, as shown in Fig. 5.3. The ALTREF Map in essence is used to track the encoder's choice on the current selected ALTREF_FRAME. It stores the virtual indices of all the ALTREF_FRAME candidates, and the virtual index associated with the current selected ALTREF_FRAME is written to the bitstream.

### 5.3.2 Multi-Layer-Multi-Reference Framework

A multi-layer framework may be constructed using the extended ALTREF_FRAMEs, and an example is given in Figure 5.4a. This framework constructs a multi-layer structure where the top layer frames are coded through the prediction from the lower layers. As discussed in Sec. 5.2.1, one GF group starts with the coding of either a KEY_FRAME or an OVERLAY frame, serving as the GOLDEN_FRAME, followed by the coding of a distant future ALTREF_FRAME candidate, denoted as ALT0 in the figure. These two frames together form the bottom layer of the




(a) Symmetric multi-reference prediction in display order


(b) Symmetric multi-reference prediction in encoding order (SE for non-filtered ALTREF_FRAMEs and O for filtered ones

Figure 5.4: An example of the symmetric multi-layer multi-reference framework.

multi-layer structure. Given a GF group, we propose to use the new coding tools to construct multi-layer structure with the following steps.

**Step 1.** Insert $k$ extended ALTREF_FRAMEs and space them equally in the GF group. Since the extended ALTREF_FRAME along with the original ALTREF_FRAME lay out the bottom layer of the hierarchy structure, they will all serve as a distant future reference. We ensure there is enough space between each frame in the bottom layer by letting

$$k = \min\left(\left\lfloor \frac{\text{length(GF)}}{4} \right\rfloor - 1, 2\right).$$

Note that due to the size constraint of the reference buffer, the maximum number of ALTREF_FRAME allowed is two.

The extended ALTREF_FRAME's divide the GF group into several subgroups. Compared to the original ALTREF_FRAME, the extended ALTREF_FRAME's are always located closer to the current coded frame, hence, a predictor of higher quality may be obtained without the use of temporal filtering. When an ALTREF_FRAME is not filtered, the "show_exsisting_frame" flag is turned on and no OVERLAY frame is added. The coding of both ALT2 and ALT1 may choose ALT0 to serve as their ALTREF_FRAME.

**Step 2.** Following coding order, the BWDREF_FRAME in each subgroup is constructed and formed the second layer from the top of the multi-layer structure. Through the virtual index manipulation, coding of the BWDREF_FRAME will use the near ALTREF_FRAME (e.g. ALT2 or ALT1) to serve as its BWDREF_FRAME and the distant ALTREF_FRAME (ALT0) to serve as its ALTREF_FRAME.

**Step 3.** The remaining frames in the GF group form the top layer of the multi-layer structure. These frames use the near future reference frame as their BWDREF_FRAME, and the next future reference frame as their ALTREF_FRAME, if available. For instance, in Figure 5.4a, all the first frames in the top layer of each subgroup have their own BWDREF_FRAME and ALTREF_FRAME explicitly coded. For those second frames in the top layer of each subgroup, through virtual index manipulation, the two available ALTREF_FRAME candidates may serve as BWDREF_FRAME and ALTREF_FRAME respectively. For instance, for Frame 6, ALT2 may serve as BWDREF_FRAME and ALT0 may serve as ALTREF_FRAME.

For the last frame in the last subgroup of the GF group, i.e. Frame 13 in the figure, `ALT0` is the only available backward reference, which may simply act as `ALTREF_FRAME` and no `BWDREF_FRAME` may be used.

Such coding structure is designed to minimize the decoding delay while to maintain a diversifying reference frame list to achieve a larger coding gain for the GF group. It is noted that the virtual index manipulation is only conducted at the encoder side, as the decoder simply identifies the virtual index associated with a specific reference frame from the bitstream. The encoder determines whether one buffered reference frame should act as `BWDREF_FRAME` or act as `ALTREF_FRAME`. We still maintain the size of reference frame buffer in the new coding tool the same as that specified in the AV1 baseline, considering the overall encoder complexity as well as the hardware design for the AV1 codec.

## 5.4 Experiment Results

In this section the experimental results of using extended reference frames are presented. The encoder adopts the proposed multi-layer framework and the results are compared against the AV1 baseline. We have tested the new approach over four different data sets, namely *low-res*, *derflr*, *medium-res*, and *hd-res*, where the first two sets contain video clips of the CIF/SIF-level resolution, the third set contains VGA-level resolution, and the last set contains HD-level resolution (e.g. 720p). The overall results are summarized in Table 5.1. The example results of

individual video clips for the *low-res* and *medium-res* are given in Table 5.3. In all cases, we simply use a VBR bitrate-controlled test condition, where videos are run at a range of target bitrates with a standard rate-control mechanism to obtain RD curves. The BDRate [30] is computed using the global PSNR as the distortion metric.

Compared against AV1 baseline, the new coding tool of the extended reference frames and the corresponding multi-layer encoder design increase the computational complexity at both the encoder and the decoder, but have a nearly negligible impact on the decoder side, as described in Table 5.2.

Table 5.1: Coding gains of the multi-layer framework using extended reference frames compared against AV1 baseline in terms of BDRate reduction over datasets of various resolutions.

| Data Set | low-res | derflr | medium-res | hd-res |
|---|---|---|---|---|
| Ext-Refs | -5.573% | -4.465% | -4.471% | -3.192% |

Table 5.2: Computational complexity increment of the proposed approach compared against AV1 baseline.

| | Encoder Side | Decoder Side |
|---|---|---|
| Ext-Refs | +74.16% | +2.12% |

Table 5.3: Coding gains of the multi-layer framework using extended reference frames compared against AV1 baseline in terms of BDRate reduction on the low and mid resolution datasets (50 video clips).

| Video | Resolution | BDRate Saving (%) | Video | Resolution | BDRate Saving (%) |
|---|---|---|---|---|---|
| akiyo | CIF | -5.789 | BQMall | 832×480 | -6.117 |
| bowing | CIF | -3.885 | BasketballDrillText | 832×480 | -3.937 |
| bridge_close | CIF | -5.908 | BasketballDrill | 832×480 | -2.970 |
| bridge_far | CIF | -6.777 | Flowervase | 832×480 | -4.109 |
| bus | CIF | -4.528 | Keiba | 832×480 | -1.274 |
| city | CIF | -5.041 | Mobisode2 | 832×480 | -2.671 |
| coastguard | CIF | -9.797 | PartyScene | 832×480 | -5.837 |
| container | CIF | -12.683 | RaceHorses | 832×480 | -1.340 |
| crew | CIF | -3.642 | aspen | 480p | -2.751 |
| flower | CIF | -13.176 | crowd_run | 480p | -11.267 |
| foreman | CIF | -4.433 | old_town_cross | 480p | -4.323 |
| harbour | CIF | -8.018 | red_kayak | 480p | 1.840 |
| highway | CIF | -2.426 | rush_field_cuts | 480p | -9.318 |
| husky | CIF | -4.256 | sintel_trailer_2k | 480p | -4.825 |
| ice | CIF | -4.308 | snow_mnt | 480p | 0.496 |
| mobile | CIF | -12.347 | speed_bag | 480p | -7.850 |
| motherdaughter | CIF | -4.794 | station2 | 480p | -2.548 |
| news | CIF | -3.214 | tears_of_steel1 | 480p | -4.122 |
| pamphlet | CIF | -1.446 | tears_of_steel2 | 480p | -6.668 |
| paris | CIF | -3.305 | touchdown_pass | 480p | -2.321 |
| signirene | CIF | -5.419 | west_wind_easy | 480p | -1.235 |
| silent | CIF | -3.380 | controlled_burn | 480p | -1.340 |
| students | CIF | -6.415 | crew | 4CIF | -2.476 |
| tempete | CIF | -9.465 | harbour | 4CIF | -8.387 |
| waterfall | CIF | -7.412 | ice | 4CIF | -2.876 |

## 5.5 Selectively Weighted Compound Prediction

In the previous sections, we introduce a new coding tool to extend the total number of reference frames from three to six to cover a wider temporal range. The increased flexibility allows the encoder to adapt to local variations. In Chapter 4, we observed that the majority of inter-predicted in video sequences are predicted using the compound mode, in which the two different predictions are combined either by simply averaging the two predictions, or by weighted averaging them according to their temporal distances to the current frame. The later method assumes that the prediction temporally closer to the current frame has less noises. It is generally true for uncompressed video sequences. However, for video compression, the predictions are from reconstructed frames rather than the actual sources. The quantization noises introduced by the encoding process can surpass temporal noises, and moreover the encoder can use different quantization levels for different frames based on current bit usage. As a result, a closer frame might have larger noises than a further frame. To address this, we introduce another variation of the compound modes wherein we allow the encoder to select the dominant prediction (out of the two predictions), which provides better prediction to current block. The encoder then combines the predictions with optimal linear coefficients. We add the proposed method as the third option of the compound modes. Even at the cost of increased overhead for selecting the compound modes and signaling the additional side information to indicate the dominant motion vector, the experi-

mental results show significant improvement, which demonstrates the effectiveness of the proposed method.

### 5.5.1 Background

In AV1 codec, compound predictions can be generated by two different approaches. Other than simply equally average the predictions from the two directions, the predictions can also be weighted based on the closeness of the reference frames to the current frame [31]. Basically, in this distance weighted compound prediction mode, it assumes the pixels along the motion trajectory can be model as the first order Markov process as shown in Fig. 5.5:

$$\begin{cases} s_l = \rho_t^{|l-m|} p_m + n_m \\ p_n = \rho_t^{|n-l|} s_l + n_l, \end{cases} \tag{5.1}$$

where $p_m$ is a pixel at reference frame $m$, $s_l$ is a pixel at current frame $l$, $\rho_t$ is the temporal correlation, and $n$ is the noise. Using this model, the optimal weights to combine the two predictions in different reference frames can be derived by solving

$$\arg\min_{w_{l,m}, w_{l,n}} E\left[(s_l - (w_{l,m} p_m + w_{l,n} p_n))^2\right].$$

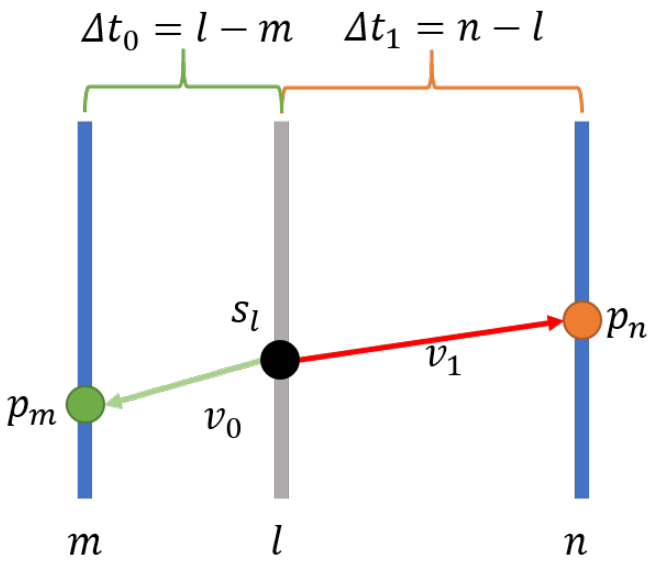



Figure 5.5: Pixels along motion trajectory can be model as first order Markov process: $s_l = \rho_t^{|l-m|} p_m + n_m$ and $p_n = \rho_t^{|n-l|} s_l + n_n$.

Assuming the noises are uncorrelated, the optimal coefficients are

$$w_{l,m} = \frac{\rho_m}{1 + \rho_m \rho_n} \quad \text{and} \quad w_{l,n} = \frac{\rho_n}{1 + \rho_m \rho_n}, \tag{5.2}$$

where $\rho_m = \rho_t^{|l-m|}$ and $\rho_n = \rho_t^{|l-n|}$.

### 5.5.2 Selectively Weighted Compound Prediction

In the previous subsection, a weight assignment method based on the first order Markov model was introduced. However, in video compression, the predictions are from reconstructed frames in which quantization noises can be introduced during the encoding process. The quantization noises are in general larger than the temporal noises at the operational points of lossy compression. Moreover,

the quantization level for each frame is not maintained the same, and is adjusted according to the frame's position in the coding structure and the current bit usage. A closer frame can have coarser quantization which leads to larger overall noises. As a result, the location of dominant prediction (which has smaller noises) cannot be determined based on closeness to the current frame, and assigning weights based on temporal distances can be inefficient. Since the encoder has access to the target frame, it can directly test which one of the two predictions has smaller noises. We propose to directly model the predictions as:

$$\begin{cases} p_m &= s_l + n_{l,m} \\ p_n &= s_l + n_{l,n}, \end{cases} \tag{5.3}$$

where $n$ is the overall noise, and $p_m$ and $p_n$ are predictions to $s_l$ in reference frame $m$ and $n$ respectively. We assume their noises, $n_{l,m}$ and $n_{l,n}$, are independent and $Laplace(\lambda_{l,m})$ distributed, whose probability density function are defined as:

$$f(x|\lambda) = \frac{\lambda}{2}\exp(-\lambda x) \tag{5.4}$$

The optimal weights to combine the predictions can be derived by solving:

$$\min\ \mathrm{E}(s - (w_m p_m + w_n p_n))^2 \tag{5.5}$$

$$s.t.\ w_m + w_n = 1,, \tag{5.6}$$

which yields

$$w_m = \frac{E(n_{l,n}^2)}{E(n_{l,m}^2) + E(n_{l,n}^2)} \quad \text{and} \quad w_n = \frac{E(n_{l,m}^2)}{E(n_{l,m}^2) + E(n_{l,n}^2)}. \tag{5.7}$$

**Mode and Weight Design**

Ideally, different coefficients that are optimal for each block can be used. However, the additional cost for signaling the coefficients can out-weight the benefit of prediction improvement. Therefore, we propose to use a one-bit indicator to signal the direction of the dominant prediction and design a fixed set of weights. Specifically, the one-bit indicator can be viewed as to rearrange the predictions such that $p_m$ in eq. (5.3) is always the dominant prediction. Note that, there are already two variations of compound mode exist in AV1 as described in Section 5.5.1. Therefore, we focus on designing the coefficients for the blocks which can benefit most from the our newly introduced mode, i.e., we focus on the set of blocks, $\mathcal{B}$, whose RD cost can be improved by the new compound mode and design weight

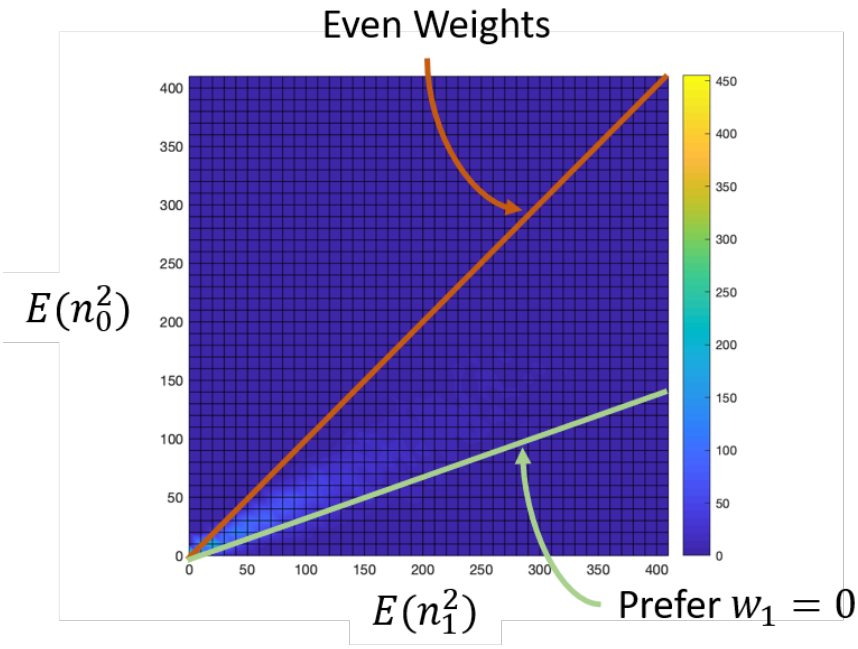
$$w^* = \arg\min_{w} \sum_{b \in \mathcal{B}} SSE_{scp}(b, w) \tag{5.8}$$

where $SSD_{scp}(b, w)$ is the sum of square prediction error of a block $b$ using the new compound mode with the weight $w$ and

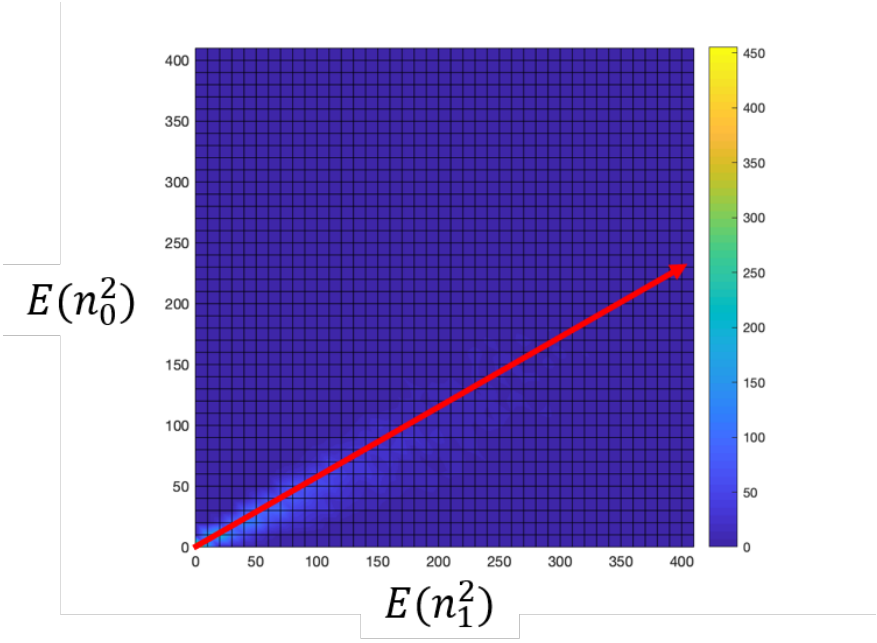
$$\mathcal{B} = \{b : SSE_{scp}(b, w') + \lambda R(\text{new compound mode}) < RD_{sel}(b)\}, \tag{5.9}$$

where $w'$ is the optimal weights for the block $b$ derived from eq.(5.7), $R$(new compound mode) $= 2$ is the rate for the new compound mode including the indicators for both the mode and the dominant prediction, and $RD_{sel}(b)$ is the RD cost of the mode the encoder selects without using the new compound mode.

Fig. 5.6a shows the distribution of blocks whose RD cost can be improved by the proposed method with optimal weight $w'$. We arrange $n_0$ such that it correspond to the noises of the predictions have smaller errors. Due to additional signaling overhead, when the optimal weights are close to even the encoder will prefer to use simple-average compound mode, or when one prediction is much better than the other the encoder prefer to use single motion vector mode. As a result, only the blocks whose the ratio of the noises within a certain range can be benefited from the design. Specifically, as shown in Fig. 5.6b, the distribution can be fitted by a line which indicates most efficient operating point (noise ratio) of the proposed method.




(a) Blocks can be benefited from the proposed method are within a certain range due to additional side information required.




(b) The noise ratio of the distribution can be fitted by a line.

Figure 5.6: Distribution of blocks whose RD cost can be improved by the proposed method with optimal weight $w^*$. $E[n_0]$ denotes the average prediction error of a block using the better prediction, and $E[n_1]$ denote the average prediction error of the other.

### 5.5.3 Experimental Result

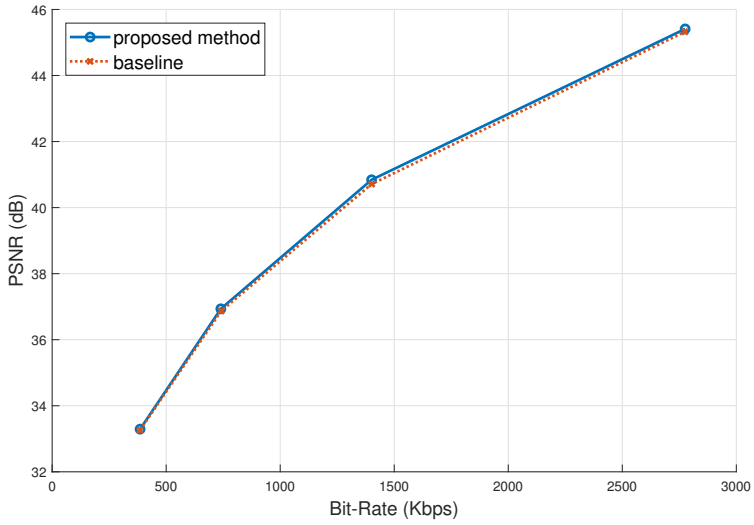
We now describe the experiments and results for the proposed method. To test the efficiency of proposed method, we encode the first 100 frames of each video sequences using target bit rate mode, wherein the encoder can adjust QP values to match the targeted bit-rate. The encoder that adopts the proposed compound mode requires additional side information to specify the particular prediction mode and the dominate prediction.

Figure 5.7 shows the coding performance of proposed method across different target bit-rates on the video sequences of *F*lower and *M*obile. Table 5.4 summarizes the BD rate reduction of all test sequences. From Figure 5.7, we can observe that the gain is consistent over all bit-rate range. Compared to the baseline, the proposed method can achieve average bit-rate reduction up to 1.5% on videos with complex textures such as *F*lower. It is because the prediction accuracy also affected by the interpolation filters which used to account sub-pixel displacement. The filters in general act as low-pass filters and will introduce additional noise, and the noise is more prevalent for the moving objects with complex texture. As a result, the Markov model is much more inaccurate. In Figure 5.8. we show the percentage of the inter-predicted area covered by a compound prediction mode. We first observe that the percentage of area covered by the distance weighted compound prediction mode increases as the bit-rate increases. It is because in the low bit-rate range, the quantization noise is more dominant than the temporal

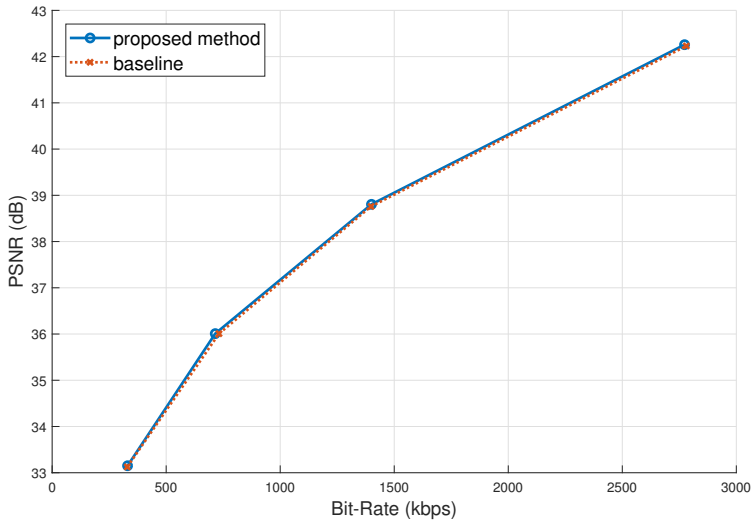
noise, rendering inefficiency for this mode. We can also observe that the area covered by the proposed method ranging from $17\% - 25\%$, which demonstrate the effectiveness of our design.

Table 5.4: BD rate reduction for the proposed new compound prediction approaches relative to AV1.

| Resolution | Sequence | Bit-Rate Reduction (%) |
|---|---|---|
| CIF | Waterfall | -0.522 |
| | Stefan | -1.072 |
| | Mobile | -1.212 |
| | Container | -0.224 |
| | Ice | -0.804 |
| | Bus | -0.350 |
| | Flower | -1.514 |
| | Coastguard | -0.253 |
| | Foreman | -1.028 |
| 416 × 240 | BlowingBubbles | -0.441 |
| | BQSquare | -0.355 |
| 832 × 480 | Keiba | -0.412 |
| | BQMall | -0.133 |
| 1080 × 720 | Shields | -0.901 |
| 1920 × 1080 | BasketballDrive | -0.476 |
| | BQTerrace | -1.593 |
| | Cactus | -0.630 |
| | Kimono1 | -0.598 |
| | Rush hour | -0.020 |
| | Average | -0.660 |




(a) *Flower*




(b) *Mobile*

Figure 5.7: Coding performance of proposed selective weighted compound prediction mode.

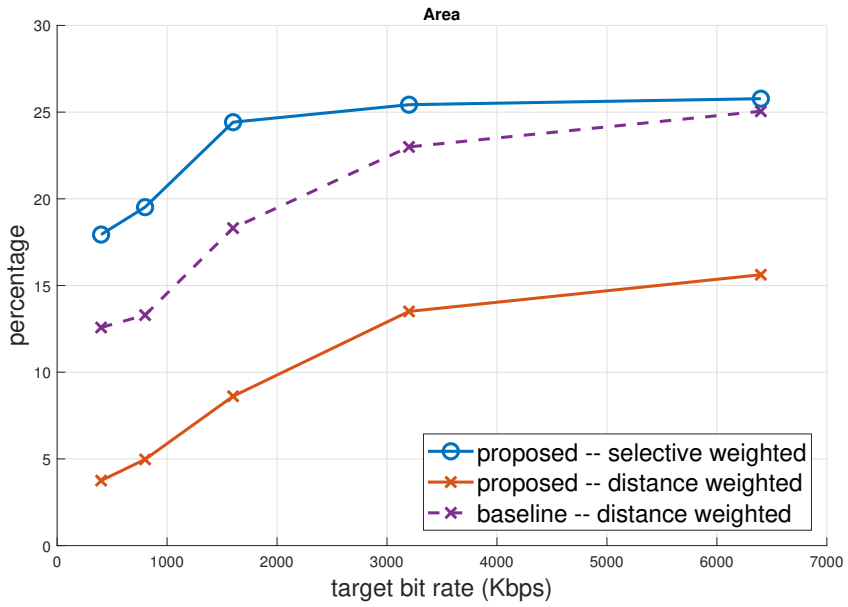



Figure 5.8: Percentage of compound prediction modes in inter-predicted area.

# Chapter 6

# Conclusion and Future Work

This dissertation primarily focuses on the optimal design of motion compensation scheme, wherein the impact of a motion vector had previously confined within in a rigid rectangular block. We propose to break such a boundary by explicitly treating motion vectors as pointers to the observation sources, and use neighboring motion vectors to form multiple estimates to a target pixel. The research is established upon this and devises several advanced techniques in an estimation-theoretic framework to efficiently utilize the motion fields and derive optimal weights to combine the estimates.

In Chapter 2, we propose a novel adaptive interpolated motion compensation (AIMC) approach, wherein the neighboring motion vectors are used as relevant estimators to the target and formulate the problem of combining multiple estimates as a linear estimation problem. The optimal linear coefficients are derived

through a training algorithm, which is developed to avoid over-fitting and to minimize the reconstruction error rather than prediction error as it is the ultimate goal for video compression. As a single set of coefficients cannot account for the varying correlation between estimators, we train $K$-sets of coefficients and allow the encoder to switch between them to adapt to local statistics.

In Chapter 3, we extend the AIMC approach in Chapter 2 to the variable block sizes setting which is widely adopted in modern video coders. By introducing an interpolation tree structure which can be inferred from the original partition tree, we enable the substantial gains from employing variable block sizes partitioning, and further improve its performance while maintaining simplicity of the original AIMC approach. A flag is added per superblock/CTU to adapt to the content of video frames to properly enable the interpolated prediction when necessary.

The non-parametric approach introduced in Chapter 2 and 3 requires additional side information to switch between interpolated coefficients to adapt to local statistics. In addition, to efficiently train and store the coefficient we are restricted to operate on the interpolation area of predefined shapes. In Chapter 4, we address these issues by introducing a parametric approach. We propose a parametric model which incorporates the Markov property for image signal and the correlations between motion vectors in the motion field generated at the encoder. As a result, the coefficients derived from the model can automatically adapt to local statistics without requiring additional side information, and we can

implement AIMC at any pixel location using any number motion vectors. Experimental results show substantial improvement over the non-parametric approach, which demonstrate the efficiency of the proposed method.

The current implementation of AIMC simply update predictions after motion vectors have been generated with the conventional block-based approach. An interesting observation is, without having encoder aware of the AIMC operation during the motion search process, the encoder can spend unnecessary bits trying to find a second motion vector (to use bi-directional/compound prediction mode) to improve the prediction accuracy which can be achieved through AIMC. Therefore, one of the future works would be to incorporate the proposed model into the motion selection loop.

In Chapter 5, we focus on optimizing the AV1 encoder, an open source video encoder founded by the Alliance of Open Media (AOM). By recognizing the disadvantage of using almost only the nearest frame as a reference frame, we first introduce a new tool to extend the number of reference frame from three to seven, which allows the reference frames cover a wider temporal range to adapt to local variations. We then design a coding structure which allocates reference frames properly to diversify the reference frame positions to maximize the efficiency of increased number of reference frames. Finally, we introduce a selectively weighted compound prediction mode which acts as a complementary role to the existing compound modes. The newly introduced mode allows the encoder to select the

dominate predictions from two directions, and combine them with the weights optimized for the blocks where the existing compound modes fall short. Simulations shows consistent performance gain over all bit-rate range.

# Bibliography


[1] *Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022*, tech. rep., Technical report, Cisco Systems Inc, 2019.

[2] "AOM - Alliance for Open Media." `http://aomedia.org/`.

[3] 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*, pp. 95991E–95991E, International Society for Optics and Photonics, 2015.

[4] 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* **22** (2012), no. 12 1649–1668.

[5] M. Wien, V. Baroncini, J. Boyce, A. Segall, and T. Suzuki, *Joint call for evidence on video compression with capability beyond hevc*, *Joint Video Exploration Team of ITU-T SG* **16** (2017).

[6] 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* **124** (2015), no. 1 44–54.

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

[8] 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* **23** (2013), no. 10 1651–1660.

[9] G. J. Sullivan and R. L. Baker, *Motion compensation for video compression using control grid interpolation*, in *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, pp. 2713–2716, IEEE, 1991.

[10] J. Bankoski, R. S. Bultje, A. Grange, Q. Gu, J. Han, J. Koleszar, D. Mukherjee, P. Wilkins, and Y. Xu, *Towards a next generation*

*open-source video codec*, in *Visual Information Processing and Communication IV*, vol. 8666, p. 866606, International Society for Optics and Photonics, 2013.

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

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

[13] 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* **1** (1992) 184–187.

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

[15] V. Verardi and C. Croux, *Robust regression in stata*, *The Stata Journal* **9** (2009), no. 3 439–453.

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

[17] A. Habibi, *Two-dimensional bayesian estimate of images*, *Proceedings of the IEEE* **60** (1972), no. 7 878–883.

[18] "WebM." `http://www.webmproject.org/`.

[19] J. Bankoski, P. Wilkins, and Y. Xu, *Technical overview of VP8, an open source video codec for the web*, in *Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2011.

[20] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, *Overview of the H.264/AVC video coding standard*, *IEEE Trans. on Circuits and System for Video Technology* **13** (2003), no. 7 560–576.

[21] T. Wiegand, X. Zhang, and B. Girod, *Motion-compensating long-term memory prediction*, *IEEE International Conference on Image Processing* **2** (1997) 53–56.

[22] T.-Y. Kuo and H.-J. Lu, *Efficient reference frame selector for H.264*, *IEEE Trans. on Circuits and System for Video Technology* **18** (2008), no. 3 400–405.

[23] V. Chellappa, P. C. Cosman, and G. M. Voelker, *Dual frame motion compensation with uneven quality assignment*, *IEEE Trans. on Circuits and System for Video Technology* **18** (2008), no. 2 249–256.

[24] D. Liu, D. Zhao, X. Ji, and W. Gao, *Dual frame motion compensation with optimal long-term reference frame selection and bit allocation*, *IEEE Trans. on Circuits and System for Video Technology* **20** (2010), no. 3 325–339.

[25] Y.-W. Huang, B.-Y. Hsieh, S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, *Analysis and complexity reduction of multiple reference frames motion estimation in H.264/AVC*, *IEEE Trans. on Circuits and System for Video Technology* **16** (2006), no. 4 507–522.

[26] M. Flierl and B. Girod, *Generalized B pictures and the draft H.264/AVC video-compression standard*, *IEEE Trans. on Circuits and System for Video Technology* **13** (2003), no. 7 587–597.

[27] A. M. Tourapis, F. Wu, and S. Li, *Direct mode coding for bipredictive slices in the H.264 standard*, *IEEE Trans. on Circuits and System for Video Technology* **15** (2005), no. 1 119–126.

[28] Y.-N. Fang, Y. Lin, and H.-J. Hsieh, *Novel direct mode decision for H.264/AVC inter B frame video coding*, *International Conference on Computing, Management and Telecommunications (ComManTel)* (2013) 198–202.

[29] M. Paul, W. Lin, C.-T. Lau, and B. S. Lee, *A long-term reference frame for hierarchical B-picture-based video coding*, *IEEE Trans. on Circuits and System for Video Technology* **24** (2014), no. 10 1729–1742.

[30] G. Bjøntegaard, *Calculation of average PSNR differences between RD curves*, *ITU-T Q.6/16 13th VCEG meeting VCEG-M33* (March, 2001).

[31] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, *et. al.*, *An overview of core coding tools in the av1 video codec*, in *2018 Picture Coding Symposium (PCS)*, pp. 41–45, IEEE, 2018.