UNIVERSITY OF CALIFORNIA
Santa Barbara

Advances in Video Coding Based on Principles of Optimal Estimation

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

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Bohan Li

Committee in charge:

Professor Kenneth Rose, Chair
Professor B.S. Manjunath
Professor Joao P. Hespanha
Onur G. Guleryuz, PhD

September 2019

The Dissertation of Bohan Li is approved.

Professor B.S. Manjunath

Professor Joao P. Hespanha

Onur G. Guleryuz, PhD

Professor Kenneth Rose, Committee Chair

June 2019

Advances in Video Coding Based on Principles of Optimal Estimation

by

Bohan Li

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. Kenneth Rose. Through the years, I have always been enlightened by his immense knowledge and sharp insights, motivated by his enthusiasm towards optimality, and encouraged by his support and patience. It would never have been such an unforgettable journey without his guidance.

This research has been funded by LG Electronics and Google and I had great internship experiences at both companies. I would like to specifically thank Dr. Onur Guleryuz, my mentor at LG and one of my doctoral committee members, for his perfect guidance into the video compression industry, and also for the memorable great moments. I would also like to thank Dr. Jingning Han, who was not only my intern mentor at Google, but also a senior alumni from our lab. His advice and help to both my research and life accompanied me through many difficulties. I'm also very thankful to Prof. Manjunath and Prof. Hespanha for serving on my doctoral committee.

I have had such a great journey in SCL. I am very grateful to my brilliant colleagues, including Tejaswi, whose patient help never disappointed, and Shunyao, who introduced me into the lab and encouraged me with her enthusiasm in pursuing her dream. I would also like to thank other lovely lab members, Sina, Wei-ting, Bharath, Ahmed, Nima and Zeyu.

Lastly, I express my deepest gratitude to my parents for their unconditional love, and to Xin, for her caring and joyful company through my most difficult times.

Curriculum Vitæ

Bohan Li

Education

2019	Ph.D. in Electrical Engineering (Expected), University of California, Santa Barbara.
2016	M.S. in Electrical Engineering, University of California, Santa Barbara.
2014	B.S. in Electronics Engineering, Tsinghua University, Beijing.

Experience

2014-2019	Graduate Student Researcher, Department of Electrical Engineering, University of California, Santa Barbara
2014-2019	Teaching Assistant, Department of Electrical Engineering, University of California, Santa Barbara
2018	Software Engineer Intern, Chrome Media Group, Google LLC., Mountain View, CA
2017	Software Engineer Intern, Chrome Media Group, Google LLC., Mountain View, CA
2016	Media Research Intern, Mobile Research Lab, LG Electronics, San Jose, CA
2015	Media Research Intern, Mobile Research Lab, LG Electronics, San Jose, CA

Awards

2017	Best Paper Award (1st place), International Conference on Image Processing (ICIP)
2017	IEEE SPS Travel Grant for ICIP 2017
2011-2012	Academic Merit Award, Tsinghua University

Publications

- B. Li, T. Nanjundaswamy and K. Rose, "Adaptive State Estimation Over Lossy Sensor Networks Fully Accounting for End-To-End Distortion," 2018 IEEE Statistical Signal Processing Workshop (SSP)

- B. Li, J. Han and Y. Xu, "Co-located Reference Frame Interpolation Using Optical Flow Estimation for Video Compression," 2018 Data Compression Conference (DCC)
- B. Li, O. G. Guleryuz, J. Ehmann and A. Vosoughi, "Layered-givens transforms: Tunable complexity, high-performance approximation of optimal non-separable transforms," 2017 IEEE International Conference on Image Processing (ICIP)
- B. Li, T. Nanjundaswamy and K. Rose, "An error-resilient video coding framework with soft reset and end-to-end distortion optimization," 2017 IEEE International Conference on Image Processing (ICIP)
- S. Li, T. Nanjundaswamy, B. Li and K. Rose, "On generalizing the estimation-theoretic framework to scalable video coding with quadtree structured block partitions," 2017 IEEE International Conference on Image Processing (ICIP)
- B. Li, T. Nanjundaswamy and K. Rose, "Block-size adaptive transform domain estimation of end-to-end distortion for error-resilient video coding," 2016 IEEE International Conference on Image Processing (ICIP)
- B. Li, L. Peng and J. Ji, "Historical Chinese Character Recognition Method Based on Style Transfer Mapping," 2014 11th IAPR International Workshop on Document Analysis Systems
- J. Ji, L. Peng and B. Li, "Graph Model Optimization Based Historical Chinese Character Segmentation Method," 2014 11th IAPR International Workshop on Document Analysis Systems
- J. Wen, B. Li, S. Li, Y. Lu and P. Tao, "Cross Segment Decoding of HEVC for Network Video Applications," 2013 20th International Packet Video Workshop

Abstract

Advances in Video Coding Based on Principles of Optimal Estimation

by

Bohan Li

This dissertation focuses on the predictive coding of video contents based on optimal prediction principles.

In the first part of the dissertation, the prediction scheme of error-resilient video coding with lossy networks is investigated. With packet-based networks suffering from potential packet loss, the prediction quality may be severely impacted not only by the loss of information, but also by the error propagation through the prediction loop. To account for such influence of lossy channels, the accurate estimation of end-to-end distortion (EED) is crucial for the encoder to perform optimal decisions. Although the recursive per-pixel optimal estimate (ROPE) and the spectral coefficientwise optimal recursive estimate (SCORE) serve as well-know solutions to optimally estimate EED in the pixel domain and the transform domain, their performance is also constrained by the incompatiblity with recent advanced coding tools. As a first step in this dissertation, the SCORE calculations are modified in order to enable the encoder to consider channel losses accurately while being able to maintain the performance improvement due to the variable

block sizes. Furthermore, it is recognized that the existing tools are not designed for lossy networks, and thus a novel framework specifically tailored for this situation is proposed with a soft-reset prediction mode. With the accurate EED estimation approach of such mode established, the encoder is able to fine-tune the error propagation and thus achieves a significant performance gain. As another example, we also establish EED estimation recursions for state estimation of wireless sensor networks and proposed an adaptive approach to account for channel errors in Kalman filter, which further proves the significance of EED estimation.

The second part of the dissertation shifts focus to the bi-directional motion compensated prediction in video coding. It is first pointed out that the conventional scheme of bi-directional motion compensation is sub-optimal, since the existing motion information among the reference frames are not efficiently utilized and the block-based motion estimation is overly crude. To overcome this issue, a novel framework with the co-located reference frame (CLRF) is proposed, where a reference frame (CLRF) is interpolated by the motion field estimated between the reference frames at the decoder, without explicit transmission of such motion field. An extra step of block-based motion estimation is then performed on top of CLRF to correct possible motion offsets. Performance gains shown by experimental results prove the effectiveness of the framework. Estimating motion field at the decoder, however, suffers from quantization error and significant complexity rise. Therefore, we then propose to apply an estimation-theory based approach

to utilize the motion vectors (MVs) already available to the decoder, and treat the associated reference pixels as observations of the current block. An optimal linear estimator is then derived and used to interpolate the CLRF. With greatly reduced complexity, this approach also provides significant coding performance improvement. The available MV candidates are then also utilized to predict the MV of the current block, in order to remove redundancy when coding MVs. Instead of a linear estimator, a novel scheme is designed to find the MV prediction that is most consistent with the MV candidates (observations) given a certain pixel correlation model. Experimental results show that the proposed scheme also achieves a boost in coding performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, the amount of video contents has seen dramatic growth and video coding applications such as video streaming, sharing and real-time conferencing are playing more and more important roles. The increasing demand of high-quality video contents and limitations in network bandwidth and storage present challenges to the video compression field. Over the years, various techniques for efficient video coding have been proposed and utilized in terms of more advanced predictive coding, transform coding, entropy coding, etc. While significant improvements in coding performance have been achieved, it should be stressed that many components of the main-stream video coding standards are still sub-optimal due to ad-hoc designs and compromises for complexity.

This dissertation focuses on predictive coding, one of the most crucial components of video compression. Prediction is widely used in nearly every video coding application in order to remove redundant information. In most recent

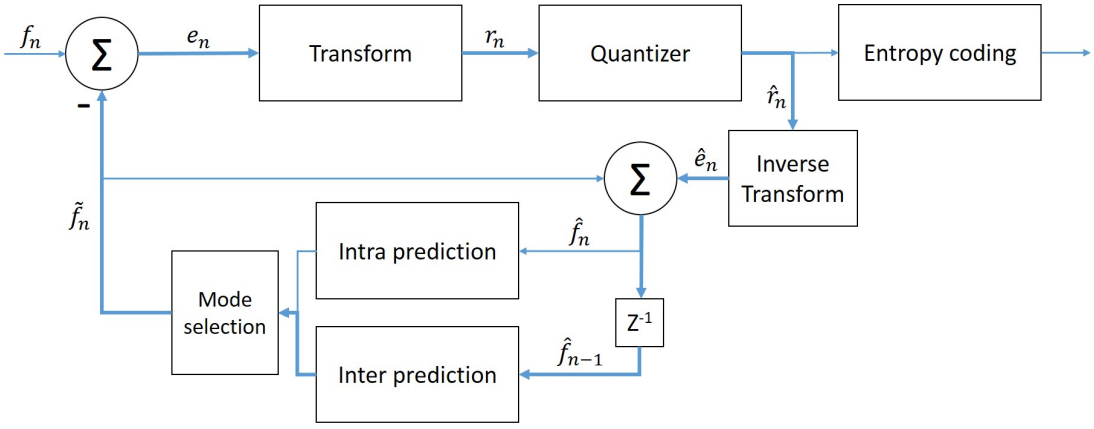


Figure 1.1: Illustration of prediction loop in a typical video coder.

video codecs, close-loop prediction is utilized, where previously reconstructed pixels are used to predict the current pixels being compressed, where intra-mode predicts from reconstructed pixels within the same frame and inter-mode from pixels in previously reconstructed frames (reference frames). As also can be seen in figure 1.1, a prediction loop exists as the result of this close-loop prediction design.

In the first part of this dissertation, the sub-optimality of prediction for lossy packet networks is investigated. Due to the prediction loop, any potential loss of information not only impacts the current packet, but also the future pixels that could predict from the lost information, causing a severe and continuous degrade in the reconstruction quality. Without considering such effect accurately, the encoder may make sub-optimal decisions even with ad-hoc error-resilient techniques such as random intra refreshing, resulting in sub-optimal coding performance. The

key, therefore, is for the encoder to accurately estimate the end-to-end distortion (EED), which is the distortion between the original video content at the encoder end and the reconstruction at the decoder end, capturing both the distortion from quantization errors and channel errors.

An established solution, the recursive optimal per-pixel estimate (ROPE), does so by tracking the first and second moments of decoder-reconstructed pixels. An alternative estimation approach, the spectral coefficient-wise optimal recursive estimate (SCORE), tracks instead moments of decoder-reconstructed transform coefficients, which enables the encoder to account for transform domain operations. However, recent advances in video coding introduce various techniques that present challenges in the EED estimation using ROPE or SCORE, limiting their performance. To demonstrate the importance of accurate EED estimation with such tools, we first propose an extended scheme with SCORE and transform domain temporal prediction (TDTP) utilizing adaptive transform block sizes. With the ability to estimate EED accurately even with the variable block size scheme, it is experimentally shown that the encoder is able to take advantage of TDTP and significant performance improvement is achieved.

Furthermore, it should be recognized that the existing tools in recent video coding applications are mostly not specifically designed for error-resilient video coding. Although with accurate EED estimation, the encoder is able to perform optimal decisions (for example, to decide between intra and inter prediction), the

lack of properly designed tool results in sub-optimal prediction quality. Therefore, we propose a novel framework that significantly expands the options available to counter error propagation by introducing optimally controlled soft resets, wherein intra and inter predictions are combined with adjustable weights to control the dependency on previous frames while accounting for the overall rate and distortion. The key aspect of the proposed framework lies in the estimation of EED, which is tackled by various extensions on top of ROPE. Experimental results show significant coding perforance improvement, proving the capacity of the proposed framework for error-resilient video coding.

Apart from video coding applications, the importance of EED estimation for communication over lossy channels in general also inspires our work on sensor networks. The EED estimation recursions similar to ROPE are developed for smart sensors networks, which utilize Kalman filters to perform state estimation. The proposed approach enables the smart sensors to optimally switch between predictive coding modes and therefore significantly improve the state estimation accuracy with limited complexity.

The next chapter of the dissertation focuses on the bi-directional prediction of video coding based on optimal estimation principles. With the hierarchical coding structure , bi-directional motion compensation approaches are widely utilized in many modern video codecs for a better coding efficiency. Conventionally, matching blocks in the two-sided reference frames are selected at the encoder, and then the

corresponding motion vectors (MVs) are transmitted to the decoder, which largely ignores the motion information already available at the decoder. Techniques to utilize such motion information have been proposed, which usually assume certain motion models that may be invalid in many scenarios.

Firstly, we propose a novel bi-directional motion compensation framework that extracts such existing motion information and interpolates a largely co-located reference frame candidate for the current frame. In the proposed approach, a dense motion field is estimated by performing optical flow estimation to capture complicated motions without any side information, while an offset motion vector is transmitted to correct possible offset from the assumed linear motion model. Various optimization schemes specifically tailored for the video coding framework are presented to further improve the performance. To account for situations where the complexity at the decoder side is constrained, a block-constrained speed-up algorithm is also proposed. Experimental results show that the proposed approach and optimization methods brings significant coding gains across a large set of video sequences. The trade-off between performance and complexity is also tested and discussed, and it is shown that our proposed speed-up algorithm is able to reduce the complexity by a large factor while still maintaining most of the performance improvements.

It should be noted that performing motion estimation at the decoder suffers from degraded reference frame quality due to quantization errors, as well as sig-

nificantly increased complexity at the decoder. Therefore, a novel bi-directional motion compensation mode is then proposed, which efficiently utilizes the motion information that is already available to the decoder (calculated at the encoder with high-quality references), without recourse to extensive search. An estimation theory based approach is proposed and utilized to provide a high quality prediction, which adaptively combines contributions from multiple motion-compensated references. Experimental results show that the proposed method, while yielding a negligible decoder side complexity, introduces a significant coding gain for a diverse set of video sequences.

Lastly, motion vector prediction is investigated, which also plays an important role in the overall performance of video codecs. With bi-directional references and previously available motion vectors, a novel motion vector prediction scheme based on estimation theory is also introduced in this part of dissertation. The approach determines a motion vector prediction that is mostly consistent with the available observations, and hence provides a better prediction than existing linear motion vector reference schemes.

Chapter 2

Error-Resilient Predictive Coding with EED Estimation

In most current video coding systems [1, 2, 3, 4], predictive coding is employed to exploit redundancies. However, due to the temporal and spatial error propagation via the prediction loop, it also increases the vulnerability to packet loss through channels. To mitigate this problem, many error resilience tools and paradigms have been employed, including forward error correction, intra refresh, multiple description coding, and macro block re-transmission [5]. These error resilience methods typically introduce redundancies in the compressed signal, and hence incur additional bit-rate costs. Therefore, the fundamental optimization problem that underlies the coder is formulated in terms of the trade-off between bit-rate and the distortion experienced at the decoder, also referred to as end-to-end distortion (EED). It is thus obvious that the encoder's ability to accurately estimate the EED, accounting for all factors including compression, packet loss and error

propagation, is crucial for the optimization of encoding decisions.

In this chapter, we first review the basic recursive formulas of recursive optimal per-pixel estimation (ROPE) [6, 7], which estimates EED in pixel domain. Recognizing the contraints of ROPE working with transform domain based techniques (an example, the transform domain temporal prediction (TDTP) [8], is also presented), the spectral coefficient-wise optimal recursive estimate (SCORE) [9] is introduced and an block-size adaptive approach is then introduced to extend SCORE to the variable block size scheme, as an illustration of the importance of accurate EED estimation in terms of utilizing efficient techniques while considering channel losses. Furthermore, a novel error-resilient video coding framework is proposed to further enable the encoder to optimize EED with fine-tuned error propagation using the soft-reset combined prediction mode. Lastly, an method of state estimation for sensor networks is derived and presented based on adaptive EED estimation with Kalman filter recursions.

2.1 Relevant Background

2.1.1 Basic Recursive Formulas of ROPE

Consider point-to-point video communication, assuming that packet loss is statistically uniformly distributed with packet loss rate p available to the encoder (for simplicity but without loss of generality, since extensions of ROPE have been

developed for different network models [10, 11] and can be generalized to the EED estimation methods introduced in this chapter). For optimal performance, the encoder must optimize its decisions with respect to the *decoder* reconstructed video quality. However, the decoder reconstruction is a random process as far as the encoder is concerned, with the influence of channel loss greatly complicated by error propagation through the prediction loop, error concealment efforts at the decoder, etc.

Therefore, ROPE considers the decoder reconstruction of each pixel as a *random variable*, and estimates the *expected* EED. Let the uncoded value of the pixel at location m in block k of frame n be denoted as $f_{n,k}^m$, and the decoder reconstruction of the pixel as $\hat{f}_{n,k}^m$. With mean squared error (MSE) distortion, the expected EED of $f_{n,k}^m$ can be formed as:

$$E\{(f_{n,k}^m - \hat{f}_{n,k}^m)^2\} = (f_{n,k}^m)^2 + 2f_{n,k}^m E\{\hat{f}_{n,k}^m\} + E\{(\hat{f}_{n,k}^m)^2\},$$

which clearly only requires the first and second moments of the decoder reconstruction $\hat{f}_{n,k}^m$.

Thus in order to accurately estimate EED, ROPE recursively tracks the moments of the decoder reconstruction. Note that different decoder error concealment methods and encoder schemes may result in different ROPE recursion formulas. In this chapter, we employ the simple 'slice copy' error concealment method,

where if the packet containing the current slice is lost, the co-located reconstruction in the previous frame is copied as the reconstruction of the current slice. We further assume each packet contains one frame for simplicity.

Two prediction modes are considered here: the intra refresh mode which predicts from reconstructed pixels within the current frame, and the inter prediction mode, which predicts from pixels in other reference frames. Note the intra refresh mode is also constrained to only predict from other pixels that also uses the intra refresh mode, and thus serves as a full refresh of error propagation, while the inter mode may introduce errors into the temporal prediction loop, resulting in less robustness to channel loss.

For the 'intra refresh' mode, since the current block is only predicted using other pixels encoded by the intra refresh mode in the same packet, as long as the packet containing the current frame is correctly received, the decoder will be able to reconstruct the current block exactly as the encoder reconstruction (denoted as $\bar{f}^m_{n,k}$). Therefore, the recursion formula to track the first and second moments are:

$$\begin{aligned} E\{\hat{f}^m_{n,k}\} &= (1-p)\bar{f}^m_{n,k} + pE\{\hat{f}^m_{n-1,k}\}, \\ E\{(\hat{f}^m_{n,k})^2\} &= (1-p)(\bar{f}^m_{n,k})^2 + pE\{(\hat{f}^m_{n-1,k})^2\} \end{aligned} \tag{2.1}$$

For the inter prediction mode, the current block k is predicted by the decoder reconstruction of another block k' in the previous frame (first-order inter pre-

diction is assumed here simply for easy presentation, without loss of generality). Denoting the quantized residual as $\hat{r}^m_{n,k}$, it can be shown that the ROPE recursions for inter prediction are:

$$\begin{aligned} E\{\hat{f}^m_{n,k}\} &= (1-p)(E\{\hat{f}^m_{n-1,k'}\} + \hat{r}^m_{n,k}) + pE\{\hat{f}^m_{n-1,k}\}, \\ E\{(\hat{f}^m_{n,k})^2\} &= (1-p)(E\{(\hat{f}^m_{n-1,k'})^2\} + 2\hat{r}^m_{n,k}E\{\hat{f}^m_{n-1,k'}\} \\ &\quad + (\hat{r}^m_{n,k})^2) + pE\{(\hat{f}^m_{n-1,k})^2\} \end{aligned} \tag{2.2}$$

As shown in (2.1) and (2.2), the first and second moments of decoder reconstructed pixels in the current frame depends only on the moments of decoder reconstructions in the previous frame, thus establishing a recursive method to track the moments, and therefore accurately estimate the EED. In practice, it is only necessary for the encoder to keep track of the first and second moments recursively, yielding minimal complexity yet optimal EED estimation.

2.1.2 Transform Domain Temporal Prediction

Since ROPE estimates the EED via pixel-domain calculations, it is inherently restricted to account for error propagation due to operations performed in the pixel domain. However, various source coding approaches of significant interest involve recursive operations in the transform domain. Particularly, estimation-theoretic approaches were proposed wherein substantial compression gains were achieved

by recursively operating in the transform domain, which is typically the discrete cosine transform (DCT) domain. For example, in [12, 13], a transform-domain based approach for scalable video coding is proposed, where the quantization levels of transform coefficients at the based layer are utilized to provide a better estimation for the enhancement layer. In this chapter we are specifically interested in another example, the transform domain temporal prediction (TDTP) [8, 14], which is used in our experiments to demonstrate the importance of transform-domain EED estimation.

Conventional motion-compensated prediction assumes that a sequence of pixels (from consecutive frames) along a motion trajectory form a temporal AR process, and that such sequences are independent of each other. This assumption clearly ignores the inter-pixel (spatial) correlation that exists among neighboring pixels in the same frame. To address such problem, instead, the TDTP approach models a pair of transform coefficients at a given frequency location m, denoted by $(x^m_{n,k}, x^m_{n-1,k'})$, of an inter-coded block k and its motion compensated reference block k' in frame n and $n-1$ respectively, as two successive samples of a scalar AR process, i.e.,

$$x^m_{n,k} = \rho_m x^m_{n-1,k'} + z^m_{n,k}, \tag{2.3}$$

where ρ_m is the correlation coefficient corresponding to that frequency, and $z^m_{n,k}$ is the innovation sequence. The advantage of such model lies in the fact that the

transform largely de-correlates the neighboring pixels inside the transform block, hence account for the spatial correlation as well as the temporal correlation.

In this model, the correlation coefficient ρ_m at each frequency coefficient can be estimated from the video statistics. The matrix of correlation coefficients for 4x4 blocks in the case of *coastguard_qcif.yuv* is provided in table 2.1 as an example. Note that the correlation is close to 1 for the DC term, but quite different otherwise, which is also observed by other video sequences with different block sizes. Since pixel-domain prediction is equivalent to constant weight for every coefficient location, such phenomenon further proves the effectiveness of TDTP, which accounts for different correlations for different coefficient locations.

Table 2.1: Matrix of correlation coefficients for the 4x4 DCT coefficients in *coastguard_qcif.yuv*

0.9998	0.9946	0.9916	0.9470
0.9893	0.9424	0.9068	0.8056
0.9807	0.9215	0.8696	0.7717
0.9680	0.9015	0.8309	0.7317

With the knowledge of the correlation coefficient matrices, TDTP employs motion-compensated prediction in a unique way. Unlike the conventional approach that applies spatial transformation on the residual pixel domain block, in TDTP, each block and its motion compensated reference are individually transformed, the DCT coefficients of the latter are weighted by frequency-appropriate correlation coefficients, and the prediction residue directly calculated in the trans-

form domain.

Note that although TDTP involves transform blocks, it has no additional requirement on the transform block size of the referenced frame (e.g, referenced block should have the same block size as the predicted block), since the motion compensated transform coefficients from the previous frame are calculated by applying the transform to the corresponding reconstructed *pixels*, thus the transform parameters in the previous frame have no impact on TDTP in the current frame. It should also be noted that since TDTP estimates the prediction in the transform domain, ROPE is not capable of accurately estimating EED with it.

2.2 Block-Size Adaptive Transform Domain EED Estimation

2.2.1 Fixed-Block-Size Transform Domain EED Estimation with TDTP Using SCORE

Transform domain based approaches, such as TDTP, inspired SCORE. In this section, we first introduce the fixed-block-size SCORE and form the basic recursive formulas.

Let the uncoded value of transform coefficient m in block k of frame n be denoted as, $x_{n,k}^{m}$, and the encoder and decoder reconstructions of the coefficient

as, $\bar{x}^m_{n,k}$, and $\hat{x}^m_{n,k}$, respectively. Let $u^m_{n,k}$ denote the uncoded value of coefficient m in this reference block[1]. Note that this reference block is possibly off-grid. The decoder reconstruction of the coefficient is denoted as $\hat{u}^m_{n,k}$. In a lossy channel, similar to ROPE, the encoder considers $\hat{x}^m_{n,k}$ and $\hat{u}^m_{n,k}$ as random variables. The expected distortion at coefficient $x^m_{n,k}$ is

$$E\{(x^m_{n,k} - \hat{x}^m_{n,k})^2\} = (x^m_{n,k})^2 - 2x^m_{n,k}E\{\hat{x}^m_{n,k}\} + E\{(\hat{x}^m_{n,k})^2\}, \tag{2.4}$$

which, similar to ROPE, clearly requires the first and second moments of the decoder reconstruction $\hat{x}^m_{n,k}$.

As also similar with ROPE, for the cases of intra refresh and inter prediction coding, SCORE deploys the following recursions to estimate the moments:

Intra refresh: The recursions are the same as in ROPE, but in the transform domain.

$$\begin{aligned} E\{\hat{x}^m_{n,k}\} &= (1-p)(\bar{x}^m_{n,k}) + pE\{\hat{x}^m_{n-1,k}\}, \\ E\{(\hat{x}^m_{n,k})^2\} &= (1-p)(\bar{x}^m_{n,k})^2 + pE\{(\hat{x}^m_{n-1,k})^2\}. \end{aligned} \tag{2.5}$$

Inter prediction with TDTP: Let $\hat{z}^m_{n,k}$ denote the quantized transform coeffi-

[1]While $u^m_{n,k}$ is indexed by n and k to indicate the location in the current frame n, it is in fact a function of pixels in frame $n-1$

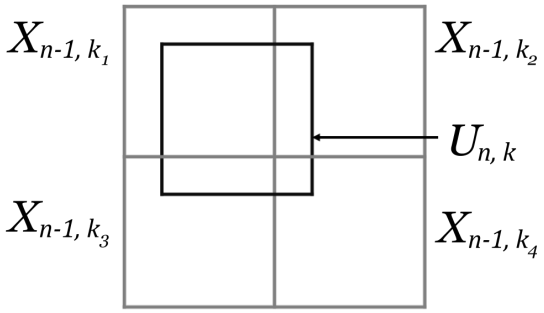


Figure 2.1: An example of a off-grid motion compensated block in the fixed block size setting.

cient residual. It can be shown that,

$$\begin{aligned} E\{\hat{x}^m_{n,k}\} &= (1-p)(\hat{z}^m_{n,k} + \rho_m E\{\hat{u}^m_{n,k}\}) + pE\{\hat{x}^m_{n-1,k}\}, \\ E\{(\hat{x}^m_{n,k})^2\} &= (1-p)((\hat{z}^m_{n,k})^2 + 2\rho_m \hat{z}^m_{n,k} E\{\hat{u}^m_{n,k}\} \\ &\quad + {\rho_m}^2 E\{(\hat{u}^m_{n,k})^2\}) + pE\{(\hat{x}^m_{n-1,k})^2\}, \end{aligned} \tag{2.6}$$

As shown in (2.6), employing TDTP in SCORE is very simple and straightforward, while basic ROPE cannot account for this transform domain method. Note that, $E\{\hat{u}^m_{n,k}\}$ and $E\{(\hat{u}^m_{n,k})^2\}$ may not be immediately available from the reference frame, since the motion compensated block could be potentially off-grid of the transform blocks in the reference frame. Thus these moments need to be calculated from the first and second moments in the reference frame.

If the block size is fixed, any off-grid block in a frame overlaps with at most four on-grid blocks (as illustrated in Fig. 2.1). Let block $U_{n,k}$ shown in the figure denote the reference block for the current block k in frame n, and it overlaps with on-grid

blocks X_{n-1,k_i} in frame $n-1$. Since DCT is a linear transformation, there exist constants $a_m^{i,l}$, named *construction constants*, such that, decoder reconstruction of block $U_{n,k}$ can be calculated as,

$$\hat{u}_{n,k}^m = \sum_{i=1}^{4} \sum_{l} a_m^{i,l} \hat{x}_{n-1,k_i}^l. \tag{2.7}$$

These constants only depend on the position of $U_{n,k}$ relative to the on-grid blocks. Given (2.7), the first and second moments of $\hat{u}_{n,k}^m$ are:

$$\begin{aligned} E\{\hat{u}_{n,k}^m\} &= \sum_{i=1}^{4} \sum_{l} a_m^{i,l} E\{\hat{x}_{n-1,k_i}^l\}; \\ E\{(\hat{u}_{n,k}^m)^2\} &= \sum_{i=1}^{4} \sum_{i'=1}^{4} \sum_{l} \sum_{l'} a_m^{i,l} a_m^{i',l'} E\{\hat{x}_{n-1,k_i}^l \hat{x}_{n-1,k_{i'}}^{l'}\}. \end{aligned} \tag{2.8}$$

Although cross-correlations are required, the advantage of operating in transform domain is that it largely decorrelates the block. Specifically, as mentioned in [9], the following assumption of 'uncorrelatedness' holds well in the DCT domain:

$$E\{\hat{x}_{n,k_i}^l \hat{x}_{n,k_{i'}}^{l'}\} \approx E\{\hat{x}_{n,k_i}^l\} E\{\hat{x}_{n,k_{i'}}^{l'}\}, \tag{2.9}$$

if $l \neq l'$, or $i \neq i'$. In this chapter, we further assume that the cross correlation between DC coefficients, $\hat{x}_{n,k_i}^0$ and $\hat{x}_{n,k_{i'}}^0$, is 1.

2.2.2 Generalization to the Variable Block Size Scheme

In the fixed block size setting in section 2.2.1, the construction constants required to estimate moments for the reference block depended only on the position of the reference block, which limited the number of such transforms. Hence they could all be calculated offline and stored to avoid addition of significant complexity to calculate them on the fly during encoding. However, in the variable block size setting, different transform block sizes could be employed along a motion trajectory. Example illustrations of such off-grid reference blocks is shown in Fig. 2.2. Similar to the fixed block size setting, due to the linearity of DCT, a set of construction constants can be obtained for each pattern to calculate the moments of $\hat{u}^m_{n,k}$ via (2.8), however, with different number of blocks and size of each block. Unfortunately, the construction constants depend not only on the position of the block $U_{n,k}$, but also on the size of the block $U_{n,k}$, and the size and position of the blocks X_{n-1,k_i} it spans. This dramatically increases the number of possible

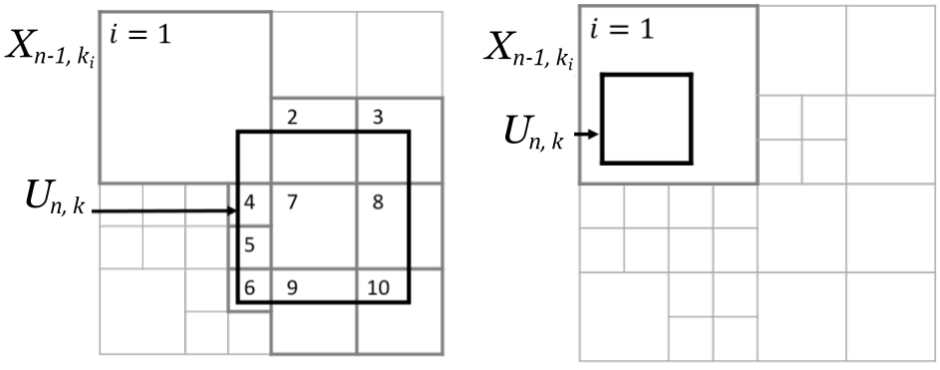


Figure 2.2: Examples of off-grid motion compensated blocks in the variable block size setting.

off-grid patterns and storing all such construction constants is impractical.

To overcome this challenge, we propose a general approach to account for any arbitrary combination of block sizes and positions, while still leveraging the advantages observed in the fixed block size setting, based on the following observations:

1. The moments of $\hat{u}^m_{n,k}$ can be derived from the moments of $\hat{x}^m_{n-1,k_i}$ if the corresponding construction constants are available;

2. The complexity of calculating and maintaining the construction constants depends on the number of possible patterns;

3. Possible patterns are very limited for the fixed block size setting.

With these observations, we propose to break the estimation of moments for the reference block into multiple steps (as shown in Fig. 2.3):

Step 1: We break the blocks X_{n-1,k_i} to a regular grid of blocks with the minimum transform block size (which is 4x4 in our experiments), and calculate the moments of the new transform coefficients. Let $Y_{n-1,k_{ij}}$ and $\hat{y}^q_{n-1,k_{ij}}$ denote the small block j and its coefficients, respectively. Again due to linearity of DCT, a set of constants $b^l_{j,q}$ exists for each coefficient q, such that,

$$\hat{y}^q_{n-1,k_{ij}} = \sum_l b^l_{j,q} \hat{x}^l_{n-1,k_i}. \tag{2.10}$$

Thus the first and second moments of $\hat{y}^q_{n-1,k_{ij}}$ are calculated via equations similar

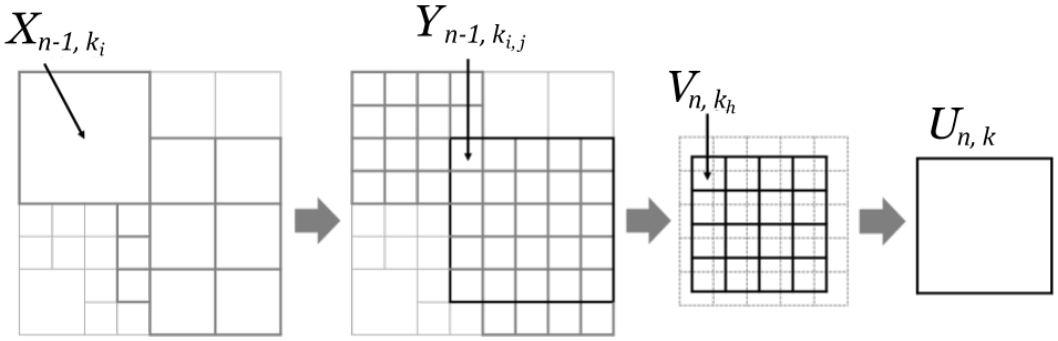


Figure 2.3: Illustration of the proposed method. For each off-grid reference block, the blocks it spans are broken into 4x4 blocks, off-grid adjusted, and finally combined to the required block size. Note that each step is an estimation of the first and second moments of the corresponding transform coefficients.

to (2.8).

Step 2: We perform motion compensation in this fixed block size setting. Given the moments of $\hat{y}^q_{n-1,k_{ij}}$, we calculate moments of the potentially off-grid coefficients according to sectionb 2.2.1. The motion compensated blocks and their coefficients are denoted as, V_{n,k_h}, and $\hat{v}^q_{n,k_h}$, respectively.

Step 3: We finally combine the small blocks V_{n,k_h} back to the required size, and calculate the moments of the new coefficients. The block we need, $U_{n,k}$, and the coefficients $\hat{u}^m_{n,k}$ can be represented in terms of $\hat{v}^q_{n,k_h}$ as:

$$\hat{u}^m_{n,k} = \sum_h \sum_f c^{h,q}_m \hat{v}^q_{n,k_h}, \tag{2.11}$$

where $c^{h,f}_m$ is another set of construction constants. Using these relations we calculate the moments of $\hat{u}^m_{n,k}$.

Overall we require three sets of construction constants, one set that is same

as section 2.2.1, and two additional sets for step 1, $b^l_{j,q}$, and step 3, $c^{h,q}_m$. However, breaking and combining blocks results in very limited patterns, leading to reduction of complexity and simplification of implementation. Note that similar to Sec. 2.2.1, we make an assumption of 'uncorrelatedness' in each step for estimation of second moments, which is shown as a valid assumption in the results.

Within the encoder, we perform step 1 right after a frame is encoded, and moments of $\hat{y}^q_{n,k_{ij}}$ are buffered for the next frame. That is, the first and second moments are maintained in a fixed block size manner, and step 2 and 3 are employed to adapt them to variable block sizes. Given the moments for $\hat{u}^m_{n,k}$, EED is estimated according to (2.6).

Note that in this section, we constrain the encoder to full-pixel motion compensation to demonstrate the potential of the proposed approach while extension of fixed-block-size SCORE to sub-pixel precision has been presented [15] and can be combined with our proposed variable block size scheme.

2.2.3 Simulation Results

In our experiments, ROPE, fixed-block-size SCORE (f-SCORE) and variable-block-size SCORE (v-SCORE) are implemented in an HEVC encoder. TDTP is employed in SCORE based coders. Slice-copy error concealment is employed in all the coders. In the fixed block size setting, the transform block size is set to 4x4, while in variable block size setting, the transform block size may

vary from 4x4 to 16x16. Note that in this section, we constrain the encoder to full-pixel motion compensation to demonstrate the potential of the proposed approach. Very recent advances in TDTP [14] extended its ability to provide considerable gains to the setting of sub pixel motion, and while the results herein are in conjunction with the original full pixel TDTP to provide the proof of concept for block size adaptive TDTP with efficient EED estimate, the approach is also extendable to be combined with sub-pixel TDTP.

We first evaluate the EED estimation accuracy of v-SCORE in coders that employ the 'random intra' error-resilience technique, where in each frame, 10% of coding units are randomly selected to be intra-coded. That is, the coders do not employ the distortion estimates for any optimization of encoding decisions. Here, TDTP is employed in both ROPE and v-SCORE coders, and we calculate the distortion in ROPE by averaging the per-pixel EED estimate, and in v-SCORE by averaging the per-coefficient estimate. We also simulate the transmission of the bitstream over 100 different realizations of a lossy channel, and average the distortion over realizations for each frame to obtain a simulation result. The ROPE and SCORE estimates are compared with the simulation result to evaluate their accuracy. The PSNRs obtained by simulation, ROPE, and v-SCORE are shown in Fig. 2.4. It can be seen that ROPE is not capable of accounting for the transform domain operations of TDTP, hence it significantly underestimates the quality at the decoder. However, v-SCORE yields fairly accurate estimation

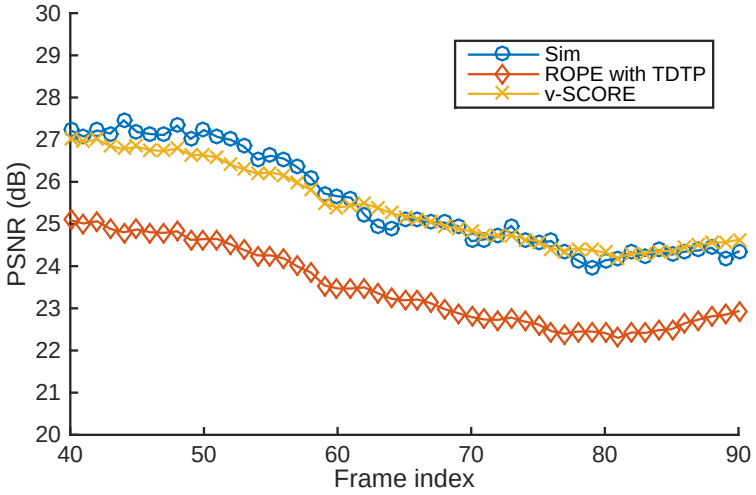


Figure 2.4: Comparison of PSNR estimates and simulated PSNR averaged over 100 simulated packet loss patterns for the BQTerrace sequence (1920x1080) at the PLR of $p = 5\%$.

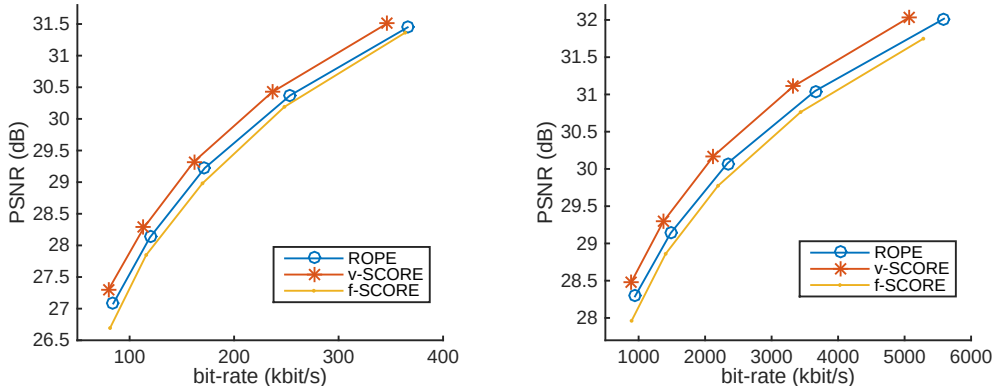
compared to the simulation result.

Next, we compare the R-D performance of employing ROPE, f-SCORE and v-SCORE for mode-selection optimization. The TDTP correlation coefficients employed were estimated via methods presented in [14], from a training set outside the test set for which results are reported. Note that TDTP is *not* employed in ROPE for this experiment. In Fig. 2.5a and Fig. 2.5b, the R-D curves at PLR of 5% for the competing coders are shown for two video sequences of different resolutions, CIF and HD. BD-PSNR improvement [16] obtained by v-SCORE, over f-SCORE (Exp v-f) and over ROPE (Exp v-R), at PLR of 1% and 5% are presented in Table 2.2. By comparing the results of v-SCORE and f-SCORE, it can be concluded that significant gains can be achieved by generalizing SCORE to the variable block size setting via the proposed approach. Note the performance

Table 2.2: BD-PSNR improvement (dB) of v-SCORE compared to f-SCORE and ROPE

	PLR $p = 1\%$		PLR $p = 5\%$	
Sequence	Exp v-f	Exp v-R	Exp v-f	Exp v-R
coastguard_cif	0.98	0.26	0.80	0.12
bus_cif	0.55	0.13	0.46	0.13
foreman_cif	0.53	0.21	0.57	0.17
flower_cif	0.39	0.37	0.23	0.17
BasketballDrive	1.16	0.02	1.17	0.18
BQTerrace	0.75	0.54	0.48	0.43
Kimono	1.25	0.03	1.02	0.07
ParkScene	0.53	0.63	0.51	0.29
Average	**0.76**	**0.27**	**0.66**	**0.20**

gain is more substantial for HD sequences, since unlike low-resolution sequences, where a 4x4 block size itself may be optimal, the block sizes may vary to a large extent for high-resolution sequences. Note that ROPE out-performing f-SCORE also demonstrates the severe limitation of fixed block size coding, where in despite employing TDTP, f-SCORE lags behind ROPE without TDTP but with variable block sizes. Finally, the gains of v-SCORE over ROPE demonstrates the utility of employing transform domain based techniques, specifically, TDTP in this chapter.



(a) foreman (CIF) (b) ParkScene (1920x1080)

Figure 2.5: R-D curves for HEVC coders with mode decision optimization via v-SCORE, f-SCORE, and ROPE, at PLR $p = 5\%$, for two sequences of different resolutions.

2.3 Error-Resilient Video Coding Framework with Soft Reset and EED Optimization

As explained in Section 2.1, with the ability to accurately estimate EED, the encoder is capable of optimally switching between the inter prediction mode, which causes error propagation through the temporal prediction loop, and the intra refresh mode, which fully stops the propagation at that instant. These two modes in effect serve merely as an on/off switch for temporal error propagation, providing a very crude control to the encoder, when in fact with accurate estimate of EED in hand, the encoder can optimally control the extent of error propagation.

Therefore, in this section, we propose an error-resilient video coding framework, where besides the inter and intra refresh modes, more options of controlling

the error propagation are allowed, and the encoder decisions are based on the EED estimation, thus providing a more flexible control over the trade-off between error-resilience and compression. Specifically, in addition to the inter mode and the intra refresh mode, the *unconstrained intra prediction* mode is first included to provide the option of allowing error propagation through the spatial prediction loop. More importantly, we propose to include the *soft-reset joint inter-intra prediction* mode in order to provide a finer control over error propagation. In the rest of this section, the above two modes and their corresponding methods to overcome the challenges of accurately estimating EED are presented.

2.3.1 Unconstrained Intra Prediction

While the constrained intra prediction (intra refresh) mode is widely used by error-resilient video coding applications, the unconstrained intra prediction mode, wherein the current block is allowed to be predicted from previously reconstructed inter-predicted pixels within the same frame, is usually the default intra prediction method in non error-resilient coders. The unconstrained intra mode is more efficient in exploiting spatial correlations between blocks, but unlike the constrained intra mode, it suffers from error propagation through the spatial prediction loop, i.e., errors in spatial neighbors of the current block (potentially through temporal error propagation) will influence its intra prediction. We introduce the unconstrained intra mode into our proposed framework as an optional mode to let the

encoder have control over the possible error propagation paths. Moreover, as to be seen in section 2.3.2, it is also part of our proposed soft reset joint prediction mode.

The EED estimation of unconstrained intra prediction is obviously different from that of constrained intra prediction shown in (2.1). If the packet containing the current frame is received, the unconstrained intra prediction $\tilde{x}^m_{n,k}(I)$ is a filtered output of previous decoder reconstructions of its neighboring blocks:

$$\tilde{x}^m_{n,k}(I) = \sum_i v_i \hat{x}^{m_i}_{n,k_i}(r), \tag{2.12}$$

where v_i are the filter coefficients. The decoder reconstruction of the ith reference, given the current frame is correctly received, is denoted as $\hat{x}^{m_i}_{n,k_i}(r)$. If e.g., this sample was reconstructed via inter prediction mode in (2.2), $\hat{x}^{m_i}_{n,k_i}(r) = \hat{x}^{m_i}_{n-1,k'_i} + \hat{r}^{m_i}_{n,k_i}$.

For the unconstrained intra mode, the moment estimation recursions can then be expressed as:

$$\begin{aligned} E\{\hat{x}^m_{n,k}\} &= (1-p)(E\{\tilde{x}^m_{n,k}(I)\} + \hat{r}^m_{n,k}) + pE\{\hat{x}^m_{n-1,k}\}, \\ E\{(\hat{x}^m_{n,k})^2\} &= (1-p)(E\{(\tilde{x}^m_{n,k}(I))^2\} + 2\hat{r}^m_{n,k}E\{\tilde{x}^m_{n,k}(I)\} \\ &\quad + (\hat{r}^m_{n,k})^2) + pE\{(\hat{x}^m_{n-1,k})^2\}. \end{aligned} \tag{2.13}$$

Note that substituting (2.12) into the second moment $E\{(\tilde{x}^m_{n,k}(I))^2\}$ shows re-

quirement of the cross correlation term $E\{\hat{x}^{m_i}_{n,k_i}(r)\hat{x}^{m_j}_{n,k_j}(r)\}$. Since only the first and second marginal moments are available, we need to approximate the spatial correlation coeffcient ρ_s. In [7], the 'exponential decay' correlation model is presented for EED estimation:

$$\rho_s(d) \approx \exp(-\alpha d), \tag{2.14}$$

where α is a parameter whose typical value is around 0.05, and d is the distance between the pixels. With this model utilized, we can now estimate the first and second moments of pixels predicted by the unconstrained intra mode, and thus estimate EED accordingly.

2.3.2 Soft Reset Joint Inter-Intra Prediction

As discussed in Section 2.3.1, the unconstrained intra prediction mode provides the encoder with an alternate error propagation path, but still does not provide the encoder with a fine control over the degree of error propagation.

To provide a controllable 'soft reset' for the error propagation, we propose to utilize the weighted average of unconstrained intra prediction and inter prediction, namely the joint inter-intra prediction. The joint prediction of pixel $f^m_{n,k}$ can be expressed as:

$$\tilde{f}^m_{n,k} = w^m(P)\tilde{f}^m_{n,k}(P) + w^m(I)\tilde{f}^m_{n,k}(I), \tag{2.15}$$

where $\tilde{f}^m_{n,k}(P)$ is the inter prediction and $\tilde{f}^m_{n,k}(I)$ is the unconstrained intra prediction, and $w^m(P)$, $w^m(I)$ are the weights for the two predictions, respectively.

Although similar joint inter-intra prediction methods (also referred to as combined inter-intra prediction) have been previously proposed [17, 18], we emphasize here that we are proposing to perform joint inter-intra prediction with both a different motivation and a different optimization approach.

First, our goal of using the joint prediction is not for a better prediction. Rather, we average intra and inter prediction in order to provide a soft reset for error propagation. Recognizing the difference in motivation, we refer to our proposed prediction mode as the 'soft reset joint prediction mode'. Also, due to the different motivations, the weights should not be targeted to address the correlation between the referenced and the predicted pixels, but should be designed for the balance between error resilience and coding efficiency.

Furthermore, since our soft-reset joint prediction is intended for video coding over a lossy channel, establishing a ROPE-like EED estimation method for the joint prediction is crucial for optimal rate-distortion (R-D) decisions in our framework. However, extending ROPE to account for the proposed mode is not trivial. To overcome the challenges, we propose the following methods to estimate EED accurately for the soft-reset joint inter-intra prediction mode.

Similar to (2.13), we first establish the moment estimation recursions as fol-

lowing:

$$
\begin{aligned}
E\{\hat{f}^m_{n,k}\} &= (1-p)(E\{\tilde{f}^m_{n,k}\} + \hat{r}^m_{n,k}) + pE\{\hat{f}^m_{n-1,k}\}, \\
E\{(\hat{f}^m_{n,k})^2\} &= (1-p)(E\{(\tilde{f}^m_{n,k})^2\} + 2\hat{r}^m_{n,k}E\{\tilde{f}^m_{n,k}\} \\
&\quad + (\hat{r}^m_{n,k})^2) + pE\{(\hat{f}^m_{n-1,k})^2\}.
\end{aligned} \tag{2.16}
$$

It is obvious that the first and second moments of the joint prediction are needed. Substuting (2.15) into the moments we have:

$$
\begin{aligned}
E\{\tilde{f}^m_{n,k}\} =& w^m(P)E\{\tilde{f}^m_{n,k}(P)\} + w^m(I)E\{\tilde{f}^m_{n,k}(I)\}, \\
E\{(\tilde{f}^m_{n,k})^2\} =& (w^m(P))^2E\{(\tilde{f}^m_{n,k}(P))^2\} \\
&+(w^m(I))^2E\{(\tilde{f}^m_{n,k}(I))^2\} \\
&+2w^m(P)w^m(I)E\{\tilde{f}^m_{n,k}(P)\tilde{f}^m_{n,k}(I)\},
\end{aligned} \tag{2.17}
$$

where the first and second moments of $\tilde{f}^m_{n,k}(I)$ are given in Section 2.3.1 and moments of $\tilde{f}^m_{n,k}(P)$ are given by moments of pixels in the previous frame.

However, note that the correlation term is still not directly available to the encoder. Moreover, the inter prediction is a reconstructed pixel in the previous frame along the motion trajectory, while the intra prediction is a reconstruction in the current frame located at the boundaries of the current block. Therefore, unlike the correlation term in Section 2.3.1, this correlation of inter and intra prediction, $E\{\tilde{f}^m_{n,k}(P)\tilde{f}^m_{n,k}(I)\}$, is actually a combination of spatial correlation and

temporal correlation.

In order to approximate the correlation term with both satisfactory accuracy and complexity, we make the 'separate correlation' assumption, wherein the correlation coefficient ρ is given by the product of the temporal correlation coefficient ρ_t and the spatial correlation coefficient ρ_s:

$$\rho(t, d) \approx \rho_t(t)\rho_s(d). \tag{2.18}$$

For the temporal correlation, the Markov model can be applied to pixels along the motion trajectory. Since first order temporal prediction is assumed, the time difference t is a constant, thus in this chapter we apply ρ_t as a constant (typically 0.95-0.98). The spatial correlation coefficient can be calculated through (2.14), where the distance d is defined as the distance from the boundary to the predicted pixel along the predicting direction for angular prediction, and the average distance to the upper and left boundary for DC and planar prediction. As will be shown in section 2.3.3, with properly set parameters, this simple approximation of cross correlation is sufficient for accurate EED estimation. With the correlation term estimated, we can finally estimate the moments in (2.17), and thus estimate the EED for the soft reset joint prediction mode.

Note that for the soft reset joint prediction mode, the unconstrained intra is used rather than intra refresh. This decision is based on the following considera-

tions. First, if the boundaries or even a portion of them are reliable (e.g., coded by the intra refresh mode), the unconstrained intra prediction portion will reasonably reset the error propagation through spatial prediction loop. Second, even when the boundaries are not reliable, the spatial error propagation path usually provides better error-resilience, which ensures the effectiveness of the joint prediction mode to serve as a soft reset. Finally, using intra refresh in joint prediction would severely constrain its ability since it would have significant bit-rate overhead due to the very limited availability of boundaries with no impact of error propagation.

It should also be noted that the purpose of adding the unconstrained intra mode and the soft reset joint prediction mode is to demonstrate the potential of our proposed error resilient video coding framework with EED estimation. The framework is general and effective even if other possible methods to control error propagation are introduced with proper EED estimation.

2.3.3 Simulation Results

In our experiments, the proposed framework and methods of EED estimation were implemented in the High Efficiency Video Coding (HEVC) [2] reference software and used for the R-D optimization of mode selection. A wide range of video sequences with resolutions ranging from 240P to 1080P were tested. For each video sequence, the first 100 frames were encoded with QP values of 27, 32, 37 and 42. The channel loss was simulated with 100 realizations at a packet loss rate

of 5%. The decoder was implemented with the simple 'slice copy' error concealment method and the video coding performance was assessed by averaging the MSE of the decoder reconstructions over the 100 realizations for each sequence and each QP.

To show the performance of our proposed error-resilient video coding framework, three sets of experiments were conducted with different availability of modes. First, the *baseline* includes only the inter and intra refresh modes. In the second set of experiments (denoted as *base+UI*), the unconstrained intra mode is enabled in addition to the two modes in baseline. Finally, in the third set (denoted as *base+UI+soft*), the proposed soft reset joint prediction mode is enabled along with the inter, intra refresh and unconstrained intra modes. For the soft reset mode in the third set, the weights of inter and intra prediction are both set to be 0.5, in order to provide a soft reset where both the predictions have the same importance.

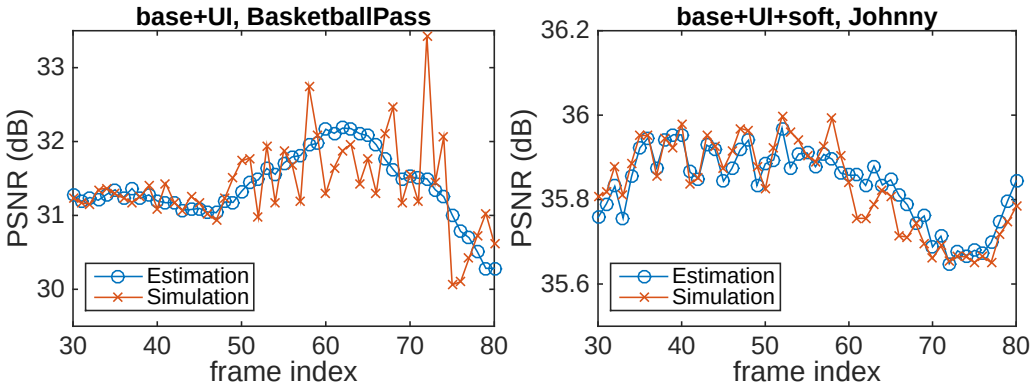


Figure 2.6: EED estimation compared with simulated ground truth.

We first show results to compare the estimated EED of each frame with the simulated EED (which can be viewed as the ground truth) to illustrate the accuracy of our proposed EED estimation methods. As shown in Fig. 2.6, EED estimation of both the base+UI and base+UI+soft settings is quite accurate and follows the general trend seen in simulated results, which confirms that the various assumptions and models introduced in section 2.3.2 are valid for our purpose.

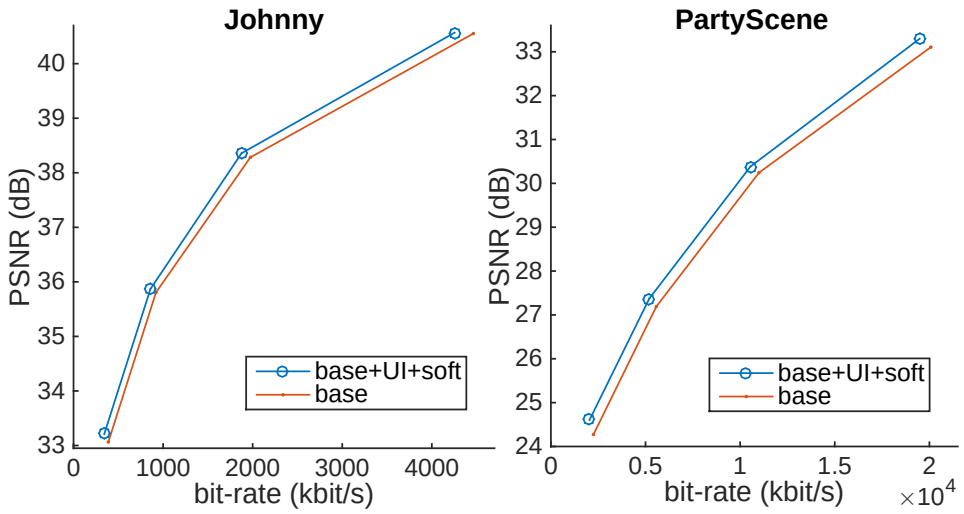


Figure 2.7: R-D curves of the proposed base+UI+soft set compared to the baseline.

Next, to demonstrate the performance of our proposed framework, compared to the baseline, the BD-PSNR [16] improvement of the base+UI set and base+UI+soft set are shown in Table 2.3. As seen in the table, with unconstrained intra mode enabled we achieve an average BD-PSNR improvement of 0.06 dB. On the one hand, this performance gain shows the potential of our framework which benefits from the ability to switch between error propagation paths while accounting for

Table 2.3: BD-PSNR improvement (dB) compared to baseline

Sequence	base+UI	base+UI+soft
mobile (CIF)	0.06	0.52
foreman (CIF)	0.06	0.24
flower (CIF)	0.01	0.15
BasketballPass (416×240)	0.09	0.16
BlowingBubbles (416×240)	0.07	0.43
PartyScene (832×480)	0.02	0.42
FourPeople (1280×720)	0.08	0.21
Johnny (1280×720)	0.18	0.27
BQTerrace (1920×1080)	0.01	0.09
ParkScene (1920×1080)	0.01	0.17
Average	**0.06**	**0.33**

the channel loss with EED estimation. On the other hand, the relatively small gain also illustrates the need for a better designed error-resilient prediction mode.

This need is confirmed by the results of the base+UI+soft set, which achieves a significant average **BD-PSNR gain of 0.33 dB** due to the introduction of the proposed soft reset joint prediction mode. The R-D curves comparing the base+UI+soft set to the baseline of two sequences are also shown in Fig. 2.7, which confirm its effectiveness for a wide range of operating points. Overall, the results show that with properly designed options, our framework provides considerable performance gain for video streaming over lossy networks.

Note in our experiments, the parameter α in (2.14), as well the value of temporal correlation coefficient ρ_t are manually selected for each sequence and are set

as constants for the test 100 frames. This is clearly suboptimal since it not only requires manual adjustment, but also ignores the fact that video content statistics are not guaranteed to be stationary, either within a single frame or across multiple frames. To address this problem, one possible future focus of research is to estimate even these parameters recursively for every pixel, which allows the encoder to capture the local statistics.

It should also be noted that, in our experiments, the weights in the soft reset joint prediction mode are chosen as a constant value of 0.5. Although the current results already show a significant performance gain, the weights should be better designed to accommodate different video content, block size, bit-rate, packet loss rate, etc. Hence, on another front, future work may focus on various approaches of designing the weights, which could be introduced as multiple options of weight combinations in the proposed framework.

2.4 Adaptive State Estimation over Lossy Sensor Networks Accounting for EED

Wireless sensor networks, wherein wireless channels are used to communicate between sensors and actuators, have been the subject of growing interest in recent years [19]. While wireless networks eliminate the need for wiring and hence are much easier to deploy in many applications, they also pose several significant chal-

lenges. A main challenge lies in the unreliable nature of wireless networks which, when combined with constrained transmitting power due to limited embedded battery life, could result in high packet loss rates and limited channel bandwidth.

Various paradigms for state estimation over unreliable channels have been proposed from a variety of perspectives. In [20], a framework was proposed to capture the effect of channel noise on conventional control systems, coupled with the corresponding stability analysis. Kalman filtering with intermittent observations (i.e., due to packet losses) is studied in [21], where convergence conditions were derived given the observation arrival probability. In [19], different lossy network models were considered and the corresponding optimal estimate was derived along with stability analysis. Moreover, the use of error correcting codes in sensor networks was studied in [22] with respect to its power efficiency.

On another front, to satisfy the data rate constraint, quantization and sampling were introduced into conventional control systems [23]. The minimum rate needed to stabilize different systems through various channels was studied in [24, 25]. To further improve the compression performance, predictive coding was employed in [26], wherein only the sign of the innovation is transmitted, while [27] proposed an extension where the innovation is coded with a multi-level quantizer.

Clearly, there is an inherent conflict between the need to reduce the data rate and the objective of better error control. On the one hand, to achieve robustness

to network errors, redundant information is usually needed which increases the rate. On the other hand, to substantially reduce the rate, one may need to employ predictive coding which exacerbates the impact of packet loss on the state estimate due to error propagation through the prediction loop, which could critically compromise performance [28].

Therefore, the tradeoff between compression efficiency and robustness to channel errors is crucial to the problem of state estimation over unreliable channels. To control this tradeoff, [29] proposes two coding modes for smart wireless sensors, where at each time instant the sensors select one of the modes according to a cost function involving the receiver's long-term average estimation error covariance and the transmission energy. To mitigate the excessive computational load due to the mode selection procedure, another solution is also proposed in [29] to provide simpler but suboptimal decisions.

In this section, a novel approach to account for the tradeoff between rate and state estimation quality is proposed. Instead of the long-term average estimation error, we proposed to consider the end-to-end distortion (EED), which is the distortion between the *encoder estimated state* and the *decoder estimated state*. Since each channel realization is not available to the encoder, we recursively estimate the mean and correlation matrix (averaging over channel realizations, not state realizations) of the EED at the encoder (sensor), leveraging a simple approach that originated from ROPE, which enables explicit rate-distortion (R-D) optimization

to decide the current optimal coding mode for a specific realization of the state process. Simulation results show that the proposed method is capable of providing considerable gains in signal-to-noise ratio of state estimation over lossy networks, given a prescribed rate constraint.

2.4.1 Problem Setup

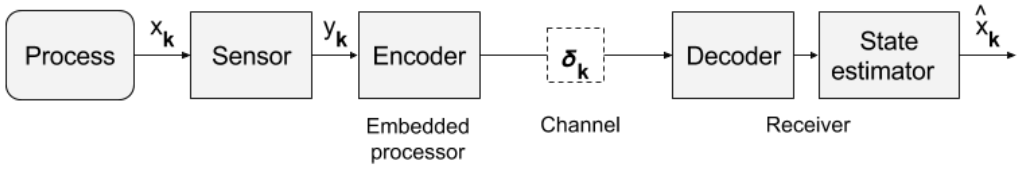


Figure 2.8: The basic setup for state estimation with a single wireless smart sensor

Consider the basic state estimation scheme shown in Fig. 2.8, with a single wireless smart sensor carrying an embedded processing unit. Let the state process be the following auto-regression (AR) process:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + \mathbf{w}_k, \tag{2.19}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector at time instant k, $A \in \mathbb{R}^{n\times n}$ is the system matrix and $\mathbf{w}_k$ is Gaussian white vector noise with covariance matrix Q.

The measurement process at the sensor is given by:

$$\mathbf{y}_k = C\mathbf{x}_k + \mathbf{v}_k, \tag{2.20}$$

where $\mathbf{y}_k \in \mathbb{R}^m$ is the observation vector at time instant k, $C \in \mathbb{R}^{m \times n}$ and $\mathbf{v}_k$ is Gaussian white vector noise with covariance matrix R.

At each time instant, the sensor generates an observation $\mathbf{y}_k$ and the processor chooses from available coding modes, denoted by t_k (details of these modes will be discussed in section 2.4.2), to encode the information and transmit the packet via a lossy channel.

In this section we consider channels with packet loss, where the model is given by a random process δ_k. $\delta_k = 0$ indicates a lost packet at time instant k, and $\delta_k = 1$ otherwise. In the setting we consider, the encoder has no feedback from the channel, and thus has no access to δ_k. It only knows the channel statistics. The receiver, on the other hand, has direct access to δ_k. For simplicity, it is assumed that δ_k is an independent and identically distributed (i.i.d.) process, noting that the proposed approach can easily be extended to more complex packet loss models. The packet loss rate is denoted as p. After decoding, the receiver calculates and outputs its state estimate $\hat{\mathbf{x}}_k$. Approaches to calculate $\hat{\mathbf{x}}_k$ naturally depend on the coding mode of the current packet, and whether or not it was received, as will be detailed in section 2.4.2.

At the encoder, different coding modes provide different tradeoffs between compression efficiency and its robustness to channel loss. How the encoder accounts for this tradeoff and determines the best mode at each time instant, given the sensor observations, is our main concern in this chapter.

From a compression perspective, this tradeoff is equivalent to achieving the best state estimation performance over a lossy network given a target rate, and is a constrained optimization problem often referred to as rate-distortion (R-D) optimization. Thus we propose to select the coding mode t_k by minimizing a Lagrangian cost function, i.e., $t_k = \arg\min_\tau J(\tau)$ where,

$$J(\tau) = E\{D_e(\tau)\} + \lambda Rate(\tau), \tag{2.21}$$

where $Rate(\tau)$ is the bitrate needed for mode τ, and λ is the Lagrangian parameter. D_e is the end-to-end distortion (EED), which is defined as the distortion between the receiver-end state estimate $\hat{\mathbf{x}}_k$ and the sensor-end state estimate $\bar{\mathbf{x}}_k$. Since the encoder does not know which packets were lost it does not know D_e exactly, hence it instead uses its best estimate, its expected value, as will shortly be explained in detail. Here, $\bar{\mathbf{x}}_k$ is given by a local steady-state Kalman filter operated at the sensor's embedded processing unit:

$$\bar{\mathbf{x}}_k = A\bar{\mathbf{x}}_{k-1} + K_s(\mathbf{y}_k - CA\bar{\mathbf{x}}_{k-1}), \tag{2.22}$$

where K_s is the steady state Kalman filter gain, given by: $K_s = P_s C^T (CP_s C^T + R)^{-1}$ and P_s is the steady state error covariance which satisfies the Riccati equation: $P_s = AP_sA^T + Q - AP_sC^T(CP_sC^T + R)^{-1}CP_sA^T$.

In (2.21), D_e is introduced to account for the state estimation error associated with channel loss and quantization error, and is the state estimation mismatch between the encoder and decoder. We adopt the mean square error as criterion: $D_e = ||\bar{\mathbf{x}}_k - \hat{\mathbf{x}}_k||_2^2$. Note that by focusing on D_e instead of the overall $D = ||\mathbf{x}_k - \hat{\mathbf{x}}_k||_2^2$ we are neglecting the estimation error $\mathbf{x}_k - \bar{\mathbf{x}}_k$ introduced at the sensor, before quantization and transmission. We will next show that by making the assumption that the sensor estimation error is uncorrelated with the state estimate mismatch between encoder and decoder, we ensure that $E\{D_e\} = E\{D\} - c$ and the R-D decision based on $E\{D_e\}$ is the same as if it were based on the expected overall state estimation error.

Lemma 1 *If the sensor estimation error $\mathbf{x}_k - \bar{\mathbf{x}}_k$ is uncorrelated with the state estimate mismatch between encoder and decoder $\bar{\mathbf{x}}_k - \hat{\mathbf{x}}_k$, then using D_e in* (2.21) *yields the same decision as using the overall state estimation error $D = ||\mathbf{x}_k - \hat{\mathbf{x}}_k||_2^2$.*

Proof: $E\{D\} = E\{||\mathbf{x}_k - \bar{\mathbf{x}}_k||_2^2\} + E\{D_e\} + E\{(\mathbf{x}_k - \bar{\mathbf{x}}_k)^T(\bar{\mathbf{x}}_k - \hat{\mathbf{x}}_k)\}$, where $E\{||\mathbf{x}_k - \bar{\mathbf{x}}_k||_2^2\}$ is a constant c given A, C, Q, R. By the assumption, the last term is 0 since $E\{(\mathbf{x}_k - \bar{\mathbf{x}}_k)\} = \mathbf{0}$. Therefore $E\{D\} = E\{D_e\} + c$ and will produce the same decision to minimize (2.21). ■

From Lemma 1, it follows that if the assumption holds approximately, then EED, while directly accounting for the channel loss for a specific realization, also represents approximately the overall state estimation performance.

Recall that we use $E\{D_e\}$ in (2.21), rather than the exact D_e. The encoder does not have access to the decoder estimated state $\hat{\mathbf{x}}_k$ and its best recourse is to estimate EED given its knowledge of the channel statistics. The estimation is given by:

$$E\{D_e(\tau)\} = E_c\{||\bar{\mathbf{x}}_k - \hat{\mathbf{x}}_k||_2^2 \mid t_k = \tau\} \tag{2.23}$$

where we use the streamlined notation $E_c\{\cdot\}$ to represent expectation conditioning on past observations and coding modes, i.e., $E_c\{\cdot\} = E\{\cdot \mid t_i, \mathbf{y}_j : i = 0, 1, ..., k-1; j = 0, 1, ..., k\}$. Since the estimation is conditioned on the sensor observations, it is still able to capture the effects of channel loss for a specific measurement from the sensor.

Note that calculating $E\{D_e(\tau)\}$ is not a simple task considering the propagation of errors from possible lost packets in the past, through the prediction loop and the Kalman filter recursions. In section 2.4.2 we will introduce an EED estimation approach which optimally estimates $E\{D_e(\tau)\}$ with a reasonable complexity via a recursive formula.

2.4.2 Coding Modes and EED Estimation

As presented in section 2.4.1, the encoder needs to obtain an EED estimate for each mode to determine the optimal via (2.21). However, due to the prediction loop introduced by some encoding modes, as well as the recursive nature of

Kalman filters, estimating EED in (2.23) is not straightforward, since the number of possible error propagation scenarios increases exponentially with the number of transmitted packets. Furthermore, a Monte Carlo simulation at the sensor is impractical due to its excessive computational load.

In this section, we introduce the encoding and decoding procedure for each mode, as well as a recursive technique to optimally estimate EED at low to moderate complexity.

Observation Mode

We first introduce the prevalent "observation" mode ($t_k = 0$), where the quantized observation $\mathbf{y}_k^q$ is sent to the decoder. The decoder generates its receiver-end state estimate by:

$$\hat{\mathbf{x}}_k = A\hat{\mathbf{x}}_{k-1} + \delta_k K_d(\mathbf{y}_k^q - CA\hat{\mathbf{x}}_{k-1}), \tag{2.24}$$

which is a steady-state Kalman filter with intermittent observations, where $K_d = P_d C^T(CP_dCC^T + R)^{-1}CP_dA^T$ and P_d satisfies the modified Riccati equation [30]: $P_d = AP_dA^T + Q - (1-p)AP_dC^T(CP_dC^T + R)^{-1}CP_dA^T$. The steady state Kalman filter accounts for the channel loss and ignores the quantization noise since it is assumed to be dominated by the observation noise and channel errors.

At the encoder, the EED estimation is given by:

$$E_c\{||\bar{\mathbf{x}}_k - \hat{\mathbf{x}}_k||_2^2\} = \bar{\mathbf{x}}_k^T\bar{\mathbf{x}}_k - 2\bar{\mathbf{x}}_k^T E_c\{\hat{\mathbf{x}}_k\} + Tr(E_c\{\hat{\mathbf{x}}_k\hat{\mathbf{x}}_k^T\}), \tag{2.25}$$

where $Tr(\cdot)$ denotes the trace. Note that the sensor state estimate $\bar{\mathbf{x}}_k$ is known to the encoder and is not considered random for the estimation.

It is evident from (2.25) that the encoder only needs to obtain the first moment $E_c\{\hat{\mathbf{x}}_k\}$ and the matrix of second moments $E_c\{\hat{\mathbf{x}}_k\hat{\mathbf{x}}_k^T\}$ (equivalently the mean vector and covariance matrix of $\hat{\mathbf{x}}_k$).

For the observation mode ($t_k = 0$), we derive the following recursive formulas to calculate the first and second moments given the known decoder procedure (2.24):

$$\begin{aligned} E_c\{\hat{\mathbf{x}}_k\} &= AE_c\{\hat{\mathbf{x}}_{k-1}\} + (1-p)K_d(\mathbf{y}_k^q - CAE_c\{\hat{\mathbf{x}}_{k-1}\}); \\ E_c\{\hat{\mathbf{x}}_k\hat{\mathbf{x}}_k^T\} &= pAE_c\{\hat{\mathbf{x}}_{k-1}\hat{\mathbf{x}}_{k-1}^T\}A^T + (1-p) \\ &\quad (FE_c\{\hat{\mathbf{x}}_{k-1}\hat{\mathbf{x}}_{k-1}^T\}F^T + K_d\mathbf{y}_k^q E_c\{\hat{\mathbf{x}}_{k-1}^T\}F^T + \\ &\quad FE_c\{\hat{\mathbf{x}}_{k-1}\}(\mathbf{y}_k^q)^T K_d^T + K_d\mathbf{y}_k^q(\mathbf{y}_k^q)^T K_d^T), \end{aligned} \tag{2.26}$$

where $F = A - K_dCA$.

As can be seen from (2.26), *only the moments from the previous time instant* $(k-1)$ are needed to calculate the moments at time k. Therefore, via the recursive update of (2.26), we can track the moments of $\hat{\mathbf{x}}_k$ and hence optimally estimate

the EED associated with the observation mode.

Innovation Mode

To achieve a better compression performance, in the innovation mode ($t_k = 1$), we choose to send the quantized innovation $\mathbf{z}_k^q$, where the innovation is $\mathbf{z}_k = \mathbf{y}_k - CA\tilde{\mathbf{x}}_{k-1}$, where $\tilde{\mathbf{x}}_k$ is the receiver-end state estimation at no channel loss, which can also be obtained at the sensor. The decoder generates the receiver-end state estimate by:

$$\hat{\mathbf{x}}_k = A\hat{\mathbf{x}}_{k-1} + \delta_k K_d \mathbf{z}_k^q. \tag{2.27}$$

Ideally, the innovation should contain all the new information from the observation $\mathbf{y}_k$. Therefore, with proper coding, the innovation mode requires a much lower rate. However, in the presence of channel loss, the encoder and decoder may "drift apart" and have different prior estimated states. This may severely compromise the usefulness of the innovation information. Hence, this mode is more heavily impacted by channel loss. The introduction of this mode largely targets the rate side of the tradeoff, and it is crucial to be able to estimate the EED to judiciously activate this mode at the encoder.

Similar to (2.26), we also establish the EED estimation procedure by recursively estimating the first and second moments of $\hat{\mathbf{x}}_k$ for the innovation mode:

$$
\begin{aligned}
E_c\{\hat{\mathbf{x}}_k\} &= AE_c\{\hat{\mathbf{x}}_{k-1}\} + (1-p)K_d\mathbf{z}_k^q; \\
E_c\{\hat{\mathbf{x}}_k\hat{\mathbf{x}}_k^T\} &= AE_c\{\hat{\mathbf{x}}_{k-1}\hat{\mathbf{x}}_{k-1}^T\}A^T + \\
&\quad (1-p)(K_d\mathbf{z}_k^q E_c\{\hat{\mathbf{x}}_{k-1}^T\}A^T + \\
&\quad AE_c\{\hat{\mathbf{x}}_{k-1}\}(\mathbf{z}_k^q)^T K_d^T + K_d\mathbf{z}_k^q(\mathbf{z}_k^q)^T K_d^T),
\end{aligned}
\tag{2.28}
$$

Here too the updates only require the first and second moments of the previous receiver-end state estimate $\hat{\mathbf{x}}_{k-1}$, enabling an efficient EED estimation for the innovation mode.

Sensing State Mode

While the innovation mode is particularly sensitive to channel loss, the observation mode also suffers from error propagation due to the recursive nature of Kalman filters. To introduce a "full reset" of the mismatch between the sensor and receiver, the sensing state mode is included, where the sensor-end state estimation $\bar{\mathbf{x}}_k$ (sensing state) is quantized and sent to the decoder. The decoder simply uses the quantized sensing state to reset its state estimation, if the packet is received:

$$
\hat{\mathbf{x}}_k = \delta_k\bar{\mathbf{x}}_k^q + (1-\delta_k)A\hat{\mathbf{x}}_{k-1}. \tag{2.29}
$$

The recursive EED estimation is straightforward:

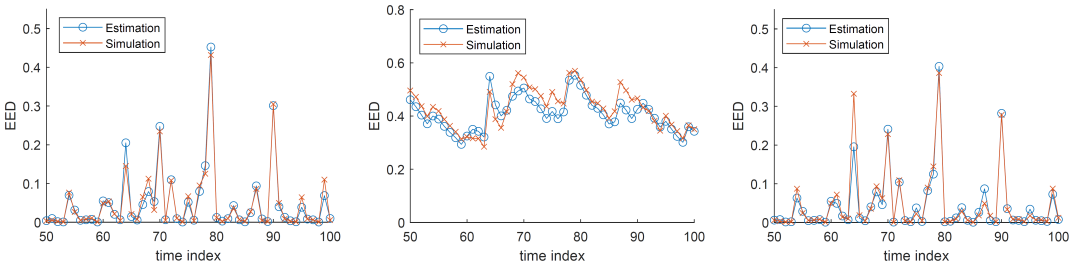
$$\begin{aligned} E_c\{\hat{\mathbf{x}}_k\} =& (1-p)\bar{\mathbf{x}}_k^q + pAE_c\{\hat{\mathbf{x}}_{k-1}\}; \\ E_c\{\hat{\mathbf{x}}_k\hat{\mathbf{x}}_k^T\} =& (1-p)\bar{\mathbf{x}}_k^q(\bar{\mathbf{x}}_k^q)^T + pAE_c\{\hat{\mathbf{x}}_{k-1}\hat{\mathbf{x}}_{k-1}^T\}A^T, \end{aligned} \tag{2.30}$$

which again only requires the first and second moments of $\hat{\mathbf{x}}_{k-1}$.

The sensing state mode stops all error propagation from previous state estimation, and thus is most robust to channel loss. However, coding the state estimation directly often requires a significantly higher rate.

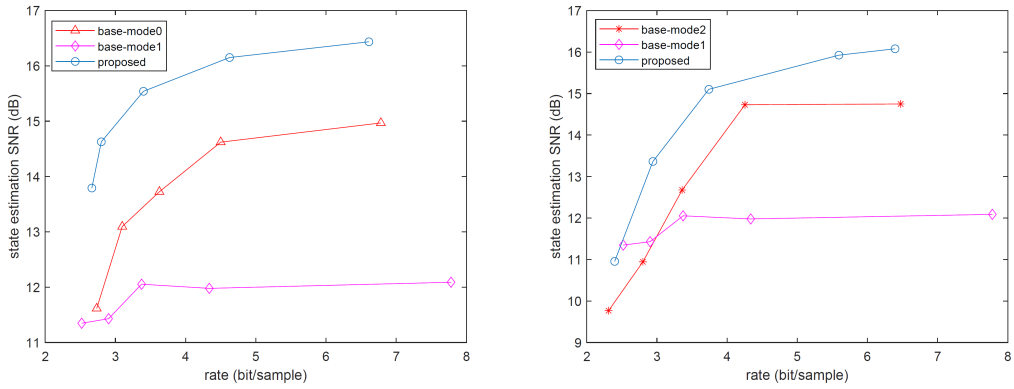
In summary, the three modes introduced in this section offer differing operating points in terms of the tradeoff between robustness to packet loss and data rate. The optimal EED estimation approach proposed here enables the encoder to minimize (2.21) to determine the optimal coding mode at each time instant.

2.4.3 Simulation Results



(a) Observation mode (b) Innovation mode (c) Sensing state mode

Figure 2.9: Comparison of simulated EED with the EED estimation by the proposed method



(a) Set 1 (b) Set 2

Figure 2.10: Rate-SNR plots of the proposed method compared to single mode coding with packet loss rate $p = 0.05$.

To achieve preliminary proof of concept and evidence for the power of the approach we consider a simple setting where the system, observation and covariance matrices are in fact scalar:

$$A = 0.96, C = 0.6, Q = 0.5, R = 0.05.$$

We assume an i.i.d. packet loss process with loss rate p.

Scalar uniform quantizers are designed for each different mode. The quantization levels are entropy coded by Huffman coding according to their probability of occurrence.

First, we evaluate the EED estimation accuracy of the proposed approach. Fig. 2.9 compares the encoder's EED estimate with actual decoder EED measurements obtained by averaging over 200 random channel realizations. The packet

loss rate is set to $p = 0.05$. It is evident that the proposed EED tracking technique provides the encoder with an accurate EED estimate for all three modes, which is critical to enable effective mode selection at the encoder.

Next, the effectiveness of the proposed mode-switching decision based on EED estimation, is presented. Here we consider two sets of comparisons. In set 1, the observation baseline (denoted by base-mode0) is given by using only the observation mode where different quantization intervals are used to generate the rate-SNR curve. The innovation baseline is also presented similarly (base-mode1). The proposed method considers both the observation and innovation modes, and switches between them according to the estimated EED and bitrate needed for each mode (side information to specify the mode is included in the rate calculation). The SNR versus rate curve is obtained by varying the Lagrange parameter λ in (2.21), where quantization intervals are separately designed for each λ. Set 2 considers a similar comparison, where the sensing state mode replaces the observation mode in both baseline (base-mode2) and the proposed method.

The rate-SNR curves are shown in Fig. 2.10. Note the SNR here is the signal-to-noise ratio of the ultimate state estimation which considers the actual state estimation distortion, instead of EED. The results reflect averaging over 200 signal realizations and 200 channel realizations with $p = 0.05$.

From the figure, it is evident that the proposed method enables the encoder to effectively switch between two different modes and considerably enhance the

ultimate state estimation for a prescribed data rate (**$\sim$2 dB** gain for set 1 and **$\sim$1 dB** gain for set 2). Moreover, this SNR gain is observed consistently across a wide range of rates, which further proves the reliability and effectiveness of the approach.

It should be emphasized that the experiments with scalar states and observations establish the proof of concept, and similar benefits are expected in the vector case, which is currently under investigation.

2.5 Conclusion

In this chapter, error-resilient predictive coding with EED estimation is investigated. A scheme to perform EED estimation in the transform domain with variable block sizes is first introduced to demonstrate the importance of accurately estimated EED.

Moreover, a novel error-resilient frame work with soft reset is proposed, which introduces the unconstrained intra mode and the soft-reset combined mode with their respective EED estimation approaches, allowing the encoder to finer control the error propagation through various paths as well as with softly-controlled strength. It is experimentally shown that the performance is significantly improved by the proposed framework.

Recognizing that state estimation of lossy sensor networks may also benefit

from such principles, EED estimation methods of various coding modes for smart sensors are also proposed. Experimental results prove that the proposed methods estimate EED accurately even with the complication of Kalman filter recursions and considerable gains are achieved.

Chapter 3

A Novel Framework of Bi-directional Motion Compensated Prediction for Video Coding

3.1 Introduction

One of the key components in video compression is the motion compensated prediction, which exploits the temporal correlation between frames to reduce coding redundancy. Conventionally, motion compensation is done by performing a block-based motion search, where a matching block in the reference frame is selected as the prediction of the current block. The associated motion vector as well as the prediction residual is then coded and transmitted to the decoder. In many recent video codecs[1, 2, 3, 4], the hierarchical coding structure is also utilized, where the video codec does not encode the frames according to the display order,

but instead in a pre-defined order with a layered coding structure. This enables the video codec to predict the current frame from not only previous frames, but also future frames that are already reconstructed before the current frame, which is referred to as the bi-directional motion compensated prediction. For block-based bi-directional motion compensation, two motion vectors, pointing from the current block to the two reference frames are calculated using block-matching algorithms (BMAs)[31, 32, 33]. The prediction is usually generated by a combination of the two reference blocks. Then the two motion vectors and the prediction residual are coded and transmitted.

However, such BMAs largely assume the movement of each pixel in the current block is identical, therefore limiting its efficacy to only tracking translational motions, while more complex motions such as rotation and scaling are beyond its capability. Although the variable block size scheme [34] utilized in recent video codecs could mitigate this incapability by dividing the current block into smaller blocks, it also introduces additional overhead.

Unlike the block-based motions utilized by BMAs, it is possible to capture complicated motions using a dense per-pixel motion field, where every pixel is associated with one motion vector. To determine such per-pixel motion fields, optical flow estimation methods are widely used. The basic optical flow estimation was proposed in [35]. Over the years, various techniques are developed to improve the optical flow estimation accuracy [36, 37, 38, 39, 40]. Recently, optical flow

estimation based on deep learning techniques are also proposed with a satisfactory performance [41, 42, 43]. Such techniques have been widely implemented in various contexts, including vision systems[44], object segmentation[45], video frame rate up-conversion (FRUC)[46], etc.

For motion compensated prediction in video coding, the per-pixel motion field can be generated by optical flow estimation in order to overcome the limits of BMAs. However, due to the much denser per-pixel motion field, it also yields a larger overhead to transmit to the decoder. In order to maintain the motion field information within a reasonable rate cost, it can be further compressed and then transmitted. In [47], the estimated dense motion field is compressed using a hierarchical finite element (HFE) representation. Alternatively in [48], the discrete cosine transform (DCT) is performed to efficiently compress the motion field. Although such methods are effective as to compress the motion field, they also introduce distortions in the motion field, degrading the overall prediction quality. To further reduce the overhead introduced by the motion field, the optical flow estimation principals are utilized in [49] to generate a block-level motion field. While it improves the BMAs by incorporating optical flow like techniques, the limits of block-based motions still persists.

Note that with the hierarchical coding structure, for the current frame, there exist reference frames at both directions, which are available not only at the encoder, but also the decoder. This suggests that at the decoder, certain motion

information is already available between the bi-directional references and can be utilized for the current frame. To exploit such motion information, in [50], block-based decoder-side motion estimation is performed between the reference frames at both the encoder and decoder and the estimated motion is projected to the current frame. The projected motion vectors are then used to generate a motion compensated prediction for the current frame. Similarly, instead of block-based motion vectors, a dense motion field can be estimated between the reference frames, and the motion compensated prediction can be generated accordingly [51, 52]. In this way it is not necessary to compress the dense motion field since no side information is needed for it to be estimated at the decoder.

With the similar intuition to exploit the motion information between the reference frames at the decoder, the Bi-directional Optical Flow (BIO) was proposed in [53, 54]. In BIO, instead of directly estimating the motion vectors at the decoder, the conventional block-based motion vectors are still calculated at the encoder and transmitted, but are then refined by performing optical flow based techniques utilizing the two-sided reference frames. The refined motion vectors are in a per-pixel fashion (forming a dense motion field), therefore are able to capture complex motions. Note that, the two conventional block-based motion vectors essentially provide a better initialization for the motion estimation/refinement process, but at the cost of extra rate redundancy, since the two motion vectors may contain motion information that is already available from the bi-directional reference frames.

It also needs to be noted that, when estimating the per-pixel motion field between the reference frames, the above approaches need to rely on certain motion models (e.g. linear motion) in order to project the motions between the reference frames to the motions of pixels in the current frame. However, in many scenarios, the actual movement may not necessarily follow the presumed motion trajectory. Therefore the prediction interpolated by such assumption may suffer from an offset from the actual source, resulting in degradation of prediction quality.

Firstly in this chapter, a novel approach to generate a *co-located reference frame* (CLRF) via optical flow estimation is proposed, which utilizes the hierarchical coding structure in a different manner. First, the optical flow between the two-sided reference frames is estimated and a per-pixel motion field is built up without any extra overhead. Then a reference frame (i.e. CLRF) is interpolated according to the motion field and the two-sided references, which should be at the same location as the current frame in the time line assuming linear motion. Note that CLRF naturally captures both translational and more complex non-translational motions due to the per-pixel motion field. Next, instead of using CLRF directly as prediction, the proposed approach treats it as an extra candidate reference frame in addition to other existing reconstructed reference frames. Regular block matching motion compensation techniques are then performed on CLRF in order to effectively compensate any potential offset in the optical flow estimation and refine the prediction quality. In addition to the basic approach,

since complexity, especially on the decoder side, is nearly always a concern for video coding applications, in section 3.3.5, we first analyzed the complexity of the proposed method as well as the impact of various parameters. Then, a block-constrained optical flow estimation algorithm is presented as a speed optimization method, which serves as an effective tool in the trade-off between coding performance and complexity. It is experimentally shown that the proposed approach achieves significant coding performance improvements, and with the proposed optimization techniques the complexity is substantially scaled down at limited expense of coding performance.

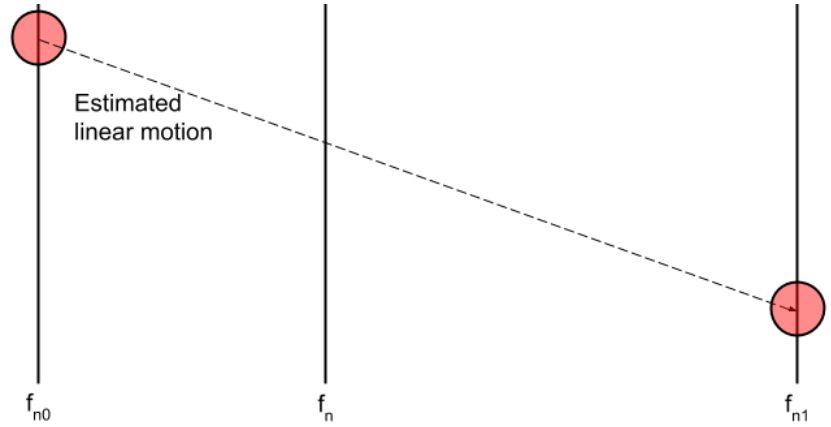
Note that besides the benefits of decoder-side motion estimation, in terms of redundancy removal, there are also disadvantages, namely, the impact of quantization error in the reference frames, and considerable increase in decoder complexity. The rest of the chapter builds on the realization that the motion information available at the decoder not only lies in the reconstructed reference frames themselves, but also in the previously transmitted explicit motion information obtained from the encoder. Such motion information may be of higher quality since the encoder has direct access to the source, uncorrupted by quantization errors. Moreover, since this information has already been decoded, there is no need for extensive motion estimation at the decoder, with the important benefit of maintaining low decoder complexity.

Therefore, we further propose the following estimation-theoretical method

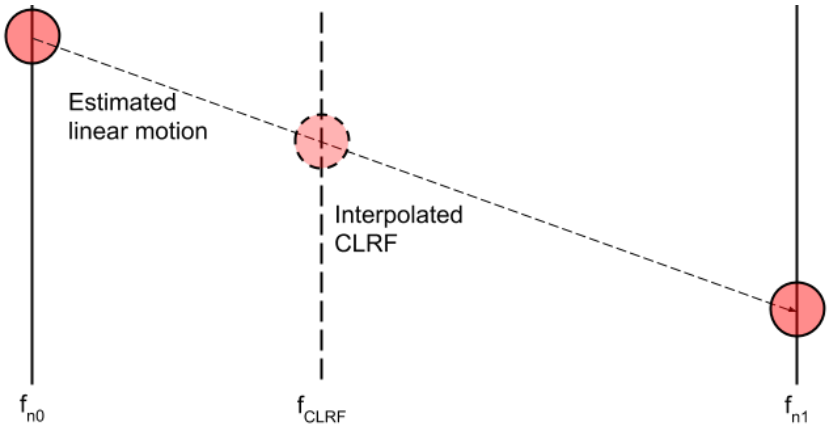
with three main steps. First, we utilize the motion vectors that were already transmitted to the decoder between the reference frames and form a set of motion vector candidates for every pixel location in the current frame. Then, from an estimation theory perspective, we treat their corresponding motion compensated references as observations that are correlated with the current pixel. An optimal linear estimator is adaptively determined using local statistics, and the prediction of the current pixel is calculated accordingly. Finally, the predictions form a CLRF and a motion vector is transmitted in order to eliminate possible offsets from the assumed linear motion. Experimental results demonstrate that without recourse to extensive motion search (i.e., at minimal complexity increase), the proposed scheme provides high quality prediction, and yields significant coding gains.

3.2 Proposed Scheme with the Co-located Reference Frame

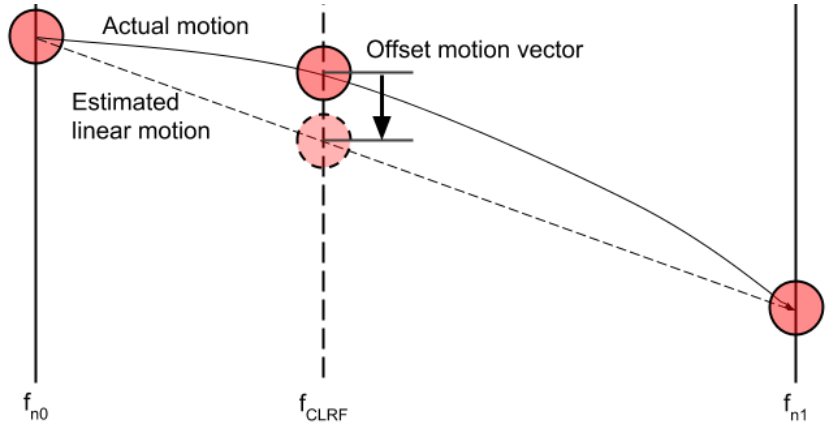
With the hierarchical coding structure, when processing frame f_n located at time n, assume there exist bi-directional reference frames f_{n_0} and f_{n_1} $(n_0 < n < n_1)$ that are previously reconstructed. Note that the reconstructed references $\hat{f}_{n_0}$ and $\hat{f}_{n_1}$ are available to both the encoder and decoder when processing f_n. As also mentioned in section 3.1, there exists motion information between the



(a) Estimate motion field



(b) Interpolate CLRF



(c) Correct offset

Figure 3.1: Illustration of the proposed prediction scheme with CLRF.

reference frames that is not fully utilized in conventional bi-directional motion compensation schemes.

To account for such motion information, as represented by the dash line in figure 3.1a, it is natural to assume linear motion between the reference frames and then estimate such motion field by performing optical flow estimation. With the estimated motion field of frame f_n, we interpolate a new frame accordingly. Since if the motion field is estimated accurately, and if the linear motion assumption is valid (i.e. objects moving with constant velocity), then the interpolated frame should be exactly co-located as f_n. Therefore we refer to the interpolated frame as the "co-located reference frame" (CLRF), as shown in figure 3.1b. The CLRF extracts the motion information between f_{n_0} and f_{n_1}, and is capable of capturing complicated non-translational motions thanks to the estimated per-pixel motion field. It is worth emphasizing that generating the CLRF does not require any extra side information.

However, it would not be ideal if the CLRF is used directly as the prediction of f_n. This is because the linear motion assumption is generally not satisfied exactly, resulting in an offset from the interpolated frame to the ground truth. This offset, even though it can be quite small, could potentially degrade the prediction quality by a large factor.

To compensate for this offset from the assumed linear motion, as illustrated in 3.1c, an *offset motion vector* is calculated and sent *per coding block* in accordance

to the block-based scheme commonly used in modern video codecs, which in our experiments, proves to be effective in correcting possible offsets. In this manner, the CLRF is treated as a regular reference frame and the offset motion vectors are treated as regular motion vectors associated with CLRF. Therefore, the integration into video codecs is quite straight-forward, and will be discussed in detail in section 3.2.1.

It should be noted that although linear motion is assumed in our implementation, one could also refer to more sophisticated motion models. However, it is still highly unlikely that the pixels will follow the model exactly, and the offset motion vectors are still needed. To adapt our proposed method to such models, one just needs to modify the way CLRF is generated to follow certain estimated trajectories, and the rest of the algorithm should stay the same.

Also note that we explain CLRF in this section based on motion field (optical flow) estimation, which is presented in detail in section 3.3. It is important to stress that the proposed scheme is a general bi-directional motion compensation scheme, and various methods to generate the co-located reference frame are possible. In section 3.4, we further introduce a novel approach based on the optimal linear estimator, which comes from a very difference angle but also shares the same basic concept of the proposed scheme using CLRF.

The overall algorithm of predicting from CLRF is provided in algorithm 1.

Algorithm 1 Overall algorithm of the proposed bi-directional prediction scheme

```
for every frame f_n do
    Determine the reference frames f_n0, f_n1
    if f_n0 or f_n1 does not exist then
        Continue to next frame
    end if
    Generate the CLRF f_CLRF
    for every block b in f_n do
        Get mv_offset associated with b
        Find block b' in f_CLRF given by mv_offset
        Set prediction b̃ ← b'
    end for
end for
```

3.2.1 Video Codec Integration

As mentioned, the CLRF is used as a candidate reference frame along with other reconstructed reference frames. Considering that the CLRF is already a blended frame, it is used only for single reference inter prediction, and is not considered for the compound reference mode (multi-reference inter prediction).

At the encoder, for every frame, first determine if the CLRF is available. It is available when there exists two reference frames in the reconstructed frame buffer, such that the current frame is in between them. If not available, regular coding scheme is used. When CLRF is available, interpolate the CLRF and used it as a candidate for single reference motion compensation.

In the bit-stream, to signal the reference frame used for a certain inter block, a flag is first sent (if CLRF is available for the current frame) to signal whether the block is using the CLRF as the reference frame. If using CLRF, the regular coding

scheme is used for the CLRF for the associated motion vector, residuals, etc. If CLRF is not used for the block, regular bit-stream syntax for other reference frame candidates will then be coded.

The decoder, same as the encoder, can also infer the availability of CLRF for every frame. If not available, regular routine follows. When the CLRF is available, the decoder also generate the CLRF and for every block, read the signaled flag first to determine its usage of CLRF reference frame. Other regular decoding schemes stay unchanged.

It is clear that by design, our proposed scheme with CLRF requires minimal change in the video codec, since the CLRF is used just as a regular reference frame. Also the same as for regular reference frames, block based motion search is also performed for the CLRF, and the selected offset motion vectors are just treated as regular motion vectors.

In this chapter, we implement the proposed approach in the AV1 codec, while the integration into other video codecs is also possible and should be fairly simple.

3.3 Co-located Reference Frame Based on Optical Flow Estimation

3.3.1 Background on the Basic Formulation of Optical Flow Estimation

The optical flow estimation is usually formed as a Lagrangian optimization, with its cost function J given by:

$$J = J_{data} + \lambda J_{spatial}, \tag{3.1}$$

where the data term, J_{data}, represents the cost of certain optical flow with respect to the data (pixel intensities) it is associated with. The spatial term $J_{spatial}$ represents the spatial constraint on the optical flow. λ is the Lagrangian parameter and controls the influence of the spatial constraint ($\lambda \geq 0$). For the rest of this section, we briefly recap the basic formulations of the two terms that are used in this chapter.

Let $I(x, y, t)$ denote the pixel intensity at location (x, y) and time instant t. For a certain pixel, let (u, v) denote its motion, where u is the horizontal component and v is the vertical component. Assuming linear motion and constant brightness

of the same object, given $0 \leq t_d \leq 1$, we have:

$$I(x - t_d u, y - t_d v, t - t_d) = I(x + (1 - t_d)u, y + (1 - t_d)v, t + (1 - t_d)). \quad (3.2)$$

This equation relates to our interested scenario where in the current frame at time t, for some pixel at (x, y) with motion vector (u, v), its predictions from the two bi-directional reference frames at $t - t_d$ and $t + (1 - t_d)$ should be located at $(x - t_d u, y - t_d v)$ and $(x + (1 - t_d)u, y + (1 - t_d)v)$.

As introduced in [35], expanding (3.2) around (x, y, t) by Taylor series expansion and eliminating higher order terms, we arrive at the following equation after reordering terms:

$$I_x u + I_y v + I_t = 0. \quad (3.3)$$

Here I_x, I_y and I_t denote the partial derivatives with respect to x, y and t. Therefore a possible choice of the data term is given by:

$$J_{data} = \sum (I_x u + I_y v + I_t)^2, \quad (3.4)$$

where the summation is over every pixel in the current frame.

As for the spatial term, the simple 4-directional 2-D Laplacian filter is consid-

ered in this chapter:

$$\Delta u_{x,y} = -4u_{x,y} + u_{x-1,y} + u_{x+1,y} + u_{x,y-1} + u_{x,y+1}, \tag{3.5}$$

and the spatial term is given by:

$$J_{spatial} = \sum \{(\Delta u)^2 + (\Delta v)^2\}. \tag{3.6}$$

Given N pixels of interest, we can form the horizontal and vertical motion components of the pixels as a $2N$ by 1 vector $\mathbf{x} = (u_0, u_1, ..., u_{N-1}, v_0, v_1, ..., v_{N-1})^T$. Then (3.4) becomes:

$$J_{data} = \mathbf{x}^T D^T D\mathbf{x} - 2\mathbf{b}_{data}^T \mathbf{x} + c_{data}, \tag{3.7}$$

where $D, \mathbf{b}_{data}$ and c_{data} can be derived from (3.4). Similarly, the spatial term (3.6) becomes:

$$J_{spatial} = \mathbf{x}^T L^T L\mathbf{x} - 2\mathbf{b}_{spat}^T \mathbf{x} + c_{spat}. \tag{3.8}$$

Here $L, \mathbf{b}_{spat}$ and c_{spat} can be derived from the Laplacian filter mask in (3.5). Note that (3.5) only defines the Laplacian filter for a general location. For the boundary pixels, different strategies will result in different $\mathbf{b}_{spat}$ and c_{spat}.

From (3.7) and (3.8), the total cost function can be written as:

$$J = \mathbf{x}^T A\mathbf{x} - 2\mathbf{b}^T\mathbf{x} + c, \tag{3.9}$$

where $A = D^T D + \lambda L^T L$, $\mathbf{b} = \mathbf{b}_{data} + \lambda \mathbf{b}_{spat}$ and $c = c_{data} + \lambda c_{spat}$.

From the definition, it is easy to see that A is a symmetric semi-positive definite matrix, thus making it a quadratic convex optimization problem. Setting the gradient of J to $\mathbf{0}$, we have:

$$A\mathbf{x} = \mathbf{b}. \tag{3.10}$$

To solve the linear equations in (3.10), due to the symmetric semi-positive definite nature of A, the conjugate gradient (CG) method [55] naturally serves as a desirable approach.

Alternatively, instead of solving the linear equations via CG, there also exist iterative approaches to optimize the cost function. For example, as described in [35], observing that the data term only involves the motion of the current pixel while the spatial term introduces the cross terms between pixels, we can estimate the spatial term of a current pixel by using the motion vectors from the last iteration to avoid such cross terms.

From (3.5), we have $\Delta u = w_c(\bar{u} - u)$ and $\Delta v = w_c(\bar{v} - v)$, where $\bar{u}$ is the

average of the horizontal components of the neighboring motion vectors (similar for $\bar{v}$ too), and w_c is the center weight of the Laplacian filter mask (for example, in (3.5), $w_c = -4$). For iteration $k+1$, this iterative approach uses the average from last iteration $\bar{u}^{(k)}$ to approximate $\bar{u}^{(k+1)}$. Since $\bar{u}^{(k)}$ is considered as a constant for iteration $k+1$, we can optimize every pixel individually. The solution at iteration $k+1$ for a certain pixel is given by:

$$\begin{aligned} u^{(k+1)} &= \bar{u}^{(k)} - \frac{I_x(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t)}{(w_c^2\lambda + I_x^2 + I_y^2)}; \\ v^{(k+1)} &= \bar{v}^{(k)} - \frac{I_y(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t)}{(w_c^2\lambda + I_x^2 + I_y^2)}; \end{aligned} \tag{3.11}$$

Note that this iterative approach is very similar to the Jacobi iterative method to solve linear equations, and is essentially a stationary iterative method, which, compared to Krylov subspace methods like CG, may suffer from slower convergence. However, with more complicated (e.g. non-linear) cost functions, such iterative approaches may yield a more straight-forward yet effective form.

In this section, we consider the CG method as our cost minimization method and stick with linear cost functions. Complexity analysis of the approach is provided in section 3.3.5.

3.3.2 Optical Flow Estimation and CLRF Interpolation

As the core of the proposed scheme, the optical flow estimation accuracy plays a crucial part in the overall prediction quality. Note that the cost optimization method introduced in section 3.3.1 depends on the assumption that the brightness of the same object stays constant, and that the spatial derivatives I_x and I_y are also stationary. However, in practice the scene is usually quite complex and such assumptions are only largely valid in a very small local area, potentially resulting in poor optical flow estimation accuracy especially with motions of large scales.

A commonly utilized technique, the *pyramid structure*, is helpful for such scenarios. With the pyramid structure, the reference frames are re-sized to different scales to form a pyramid, and the optical flow is first calculated for the smaller scale to initialize the optical flow estimation of the next level (with a larger scale). In this way, it is easier to capture large motions at the smaller scale, while at larger scales the details are further refined.

Observing that re-sizing the pixels potentially could lose details, and also that calculating the derivatives at a small scale increases the chance that the derivative filter mask spanning out of the stationary local areas, in this chapter we use a slightly different approach for the pyramid structure. Instead of re-sizing the pixels, we first calculate the derivatives at the original scale, and then re-size the derivatives to the desired scale.

In addition, at each pyramid level, multiple *warping steps* are also performed

to further improve the estimation accuracy. After calculating the optical flow at a certain step, we warp the reference frames accordingly towards the current frame f_n. Then at the next step, the motion field is updated by estimating the optical flow between the warped reference frames, and will be used for future warping steps. In this way noises in derivative calculation can be gradually attenuated, and the motion field is refined at each warping step.

To interpolate the CLRF, we warp the two reference frame according to the final optical flow estimated, and then blend them with the weighted average of the warped references:

$$I_{CLRF}(x,y) = (1-t_d)I_{n_0}(x_{n_0}, y_{n_0}) + t_d I_{n_1}(x_{n_1}, y_{n_1}), \tag{3.12}$$

where $I_{CLRF}(x,y)$, $I_{n_0}(x,y)$ and $I_{n_1}(x,y)$ denote the pixel intensity at location (x,y) in f_{CLRF}, f_{n_0} and f_{n_1} respectively. The relative ratio t_d is defined as $t_d = (n-n_0)/(n_1-n_0)$. Denoting the motion vector associated with location (x,y) in the CLRF frame as (u,v), then $x_{n_0} = x - t_d u$, $x_{n_1} = x + (1-t_d)u$, and $y_{n_0} = y - t_d v$, $y_{n_1} = y + (1-t_d)v$.

3.3.3 Motion Field Initialization

The optical flow estimation, as also stated in section 3.3.2, relies on the pixel intensity constancy condition. In practice such condition is only largely valid

in local areas for a short time interval. Therefore, when the motion vector for a certain object is of a larger scale than the local area, it would be very hard to estimate the optical flow since the intensity constancy condition is violated. When performing optical flow estimation for our proposed scheme, such scenario frequently appears due to dramatic motions in the video content. Also, if the two-sided reference frames are far apart in the hierarchical coding structure, the resulting motion between the reference frames is also often quite large.

The above problem can be mitigated by applying a better initialization of the motion field, with which the regions pointed by the initialized motion already belong to the same local area and the intensity constancy condition is satisfied. Thus performing optical flow estimation on top of such initialization yields much higher accuracy.

In many optical flow estimation applications, motion search is performed between the interested frames first to provide a good initialization. However, it is not favored for our proposed method considering the facts that: 1) motion search at the decoder adds much complexity and is not a desirable approach for video coding; 2) different from many other applications where just the two reference frame are available, in video coding, with our scheme, the two reference frames are already analyzed by the encoder and useful information may have already been extracted and buffered for the current frame.

Therefore, we instead propose the following initialization method which does

not engage extensive motion search. instead, it utilizes the motion vectors associated with the reference frames which are already available for both the encoder and decoder.

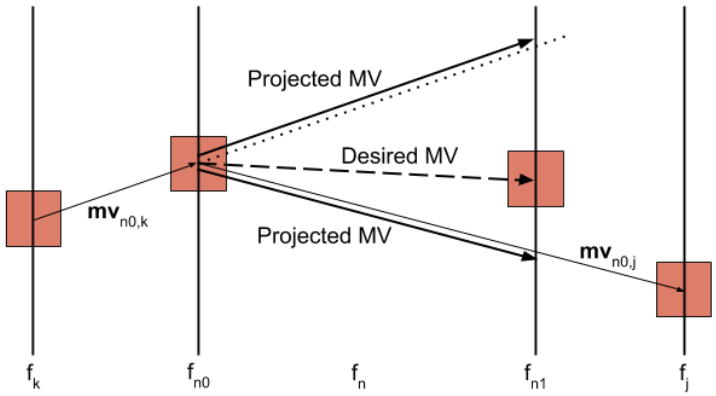
Still consider the scenario with the current frame f_n and the two-sided reference frames f_{n_0} and f_{n_1}. First, since the initialization essentially tries to coarsely estimate the motion between f_{n_0} and f_{n_1}, we look at every inter-predicted block in the two reference frames, and if the associated motion vector for the block points to the other reference frame, then this motion vector is considered as part of the initialization of the motion field. We refer to such motion vectors that are directly obtained between the reference frames as the *direct MVs*.

Although the direct MVs serve as a high quality initialization of the motion field, there is the possibility that some blocks in the reference frames f_{n_0} and f_{n_1} do not refer from the other reference frame, resulting in many uninitialized regions in the current frame.

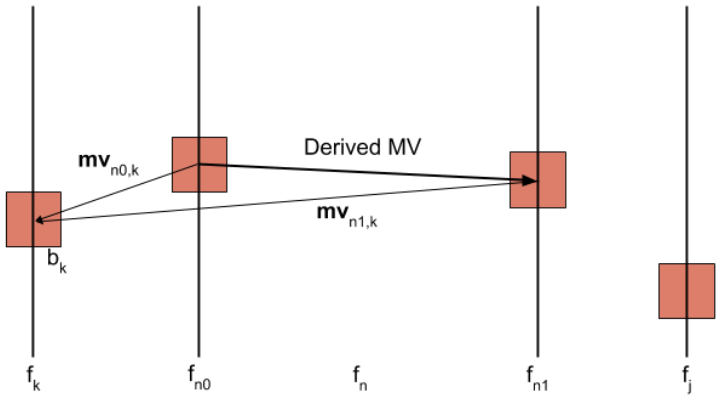
In [56], a method based on linear projection (developed from a motion vector reference scheme proposed in [57]) is used to reduce the number of uninitialized regions. As shown in figure 3.2a, taking f_{n_0} as an example, if the motion vector of a certain block in f_{n_0} does not point to f_{n_1}, we project this motion vector to f_{n_1} by assuming linear motion. Such motion vectors are referred to as *projected MVs*.

However, although the projected MVs may fill in the uninitialized regions, the

quality of them may not be satisfactory when the motion is not linear (note how the projected MVs differ from the desired initialization in figure 3.2a).



(a) Calculation of the projected MVs



(b) Calculation of the proposed derived MVs

Figure 3.2: Examples of different methods to calculate the motion field initialization. For a certain block in f_{n_0}, if its associated motion vector does not point to f_{n_1}, but other frames f_k or f_j, we can calculate the projected MVs or the proposed derived MVs, and use them as initialization. Note how the proposed derived MVs in figure 3.2b serve as a better initialization when the motion is non-linear.

Recognizing this problem, in this section, we propose another approach to provide more reliable alternative initialization in addition to the direct MVs. Figure 3.2b illustrates how this approach works. Still taking f_{n_0} as an example, if the mo-

tion vector of a certain block, $\mathbf{mv}_{n_0,k}$, is not pointing to f_{n_1}, but block b_k in frame f_k, we check whether there exists any motion vector $\mathbf{mv}_{n_1,k}$ that lies between f_{n_1} and f_k, pointing to/from b_k. If such motion vector exists, then the difference (or sum, depending on the direction of $\mathbf{mv}_{n_1,k}$) of the two motion vectors represents the motion between f_{n_0} and f_{n_1}, and thus can be used as initialization. We refer to these motion vectors as the *derived MVs*. As can be seen from 3.2b, the derived MVs do not rely on the linear motion assumption, and hence are more reliable than the projected MVs.

In our proposed scheme, we use the direct MVs together with the derived MVs to initialize our motion field. As such a motion vector crosses the current frame, we find the nearest pixel to the crossing location and assign the motion vector as the initialization of the motion field for this pixel. The remaining uninitialized regions are filled by copying the initialized motion vector of the nearest available pixel.

In conclusion, we recognize that motion search and selection are already performed by the video coder, and thus utilize the information already extracted to provide a high quality motion field initialization without extensive complexity cost.

3.3.4 MV Prediction with CLRF and Offset MVs

In section 3.2.1, we proposed to integrate the CLRF into video codecs as a regular reference frame, and also regard its associated offset motion vectors as regular motion vectors. However, note that the offset motion vectors, though treated as regular, do not represent the motion of the block, but rather the offset from the assumed linear motion.

In this section, we point out that such difference in physical meaning between the offset motion vector and the regular motion vectors could result in problems for the motion vector prediction scheme, and provide a novel approach to perform motion vector prediction with CLRF to overcome these issues.

In video coding, the motion vector of every inter-predicted block is transmitted to the decoder. To improve the coding efficiency, the motion vectors are first predicted and the prediction residuals are then coded. In many video codecs, the motion vectors of the spatial and temporal neighboring blocks are used as the prediction. The motion field initialization introduced in section 3.3.3 can also serve as a temporal prediction of motion vectors.

However, with our proposed scheme with CLRFs, the motion vector we predict from could be an offset motion vector associated to a CLRF, while the motion vector to be predicted is a regular motion vector. As we have explained, these two motion vectors represents different physical meaning, and hence could result in low-quality prediction, potentially breaking the motion vector prediction loop.

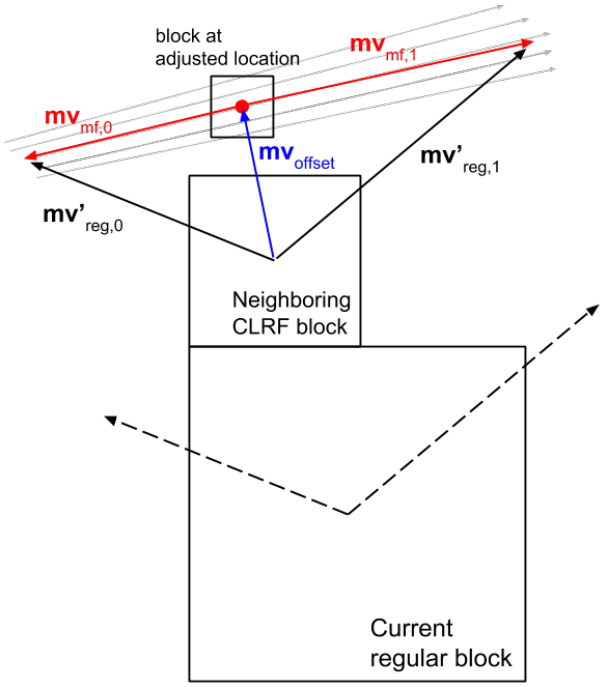


Figure 3.3: Predicting regular motion vectors from an offset motion vector.

To solve this problem, notice that although the offset motion vector do not represent the actual motion, together with the estimated optical flow, the actual motion can be derived for that neighboring block. As shown in figure 3.3, which takes the spatial motion vector prediction as an example (assuming the CLRF and the current block share the same reference frames), the derivation is done by the following steps:

First, find the adjusted location given by the location of the neighboring CLRF block and its offset motion vector $\mathbf{mv}_{offset}$. At the adjusted location, find its motion field given by the optical flow estimation. By averaging the motion field in the adjusted region, form the motion vectors pointing from this adjusted location to

f_{n_0} and f_{n_1}, denoted as $\mathbf{mv}_{mf,0}$ and $\mathbf{mv}_{mf,1}$. Lastly, the motion vector predictions are given by concatenating $\mathbf{mv}_{offset}$ with $\mathbf{mv}_{mf,0}$ and $\mathbf{mv}_{mf,1}$.

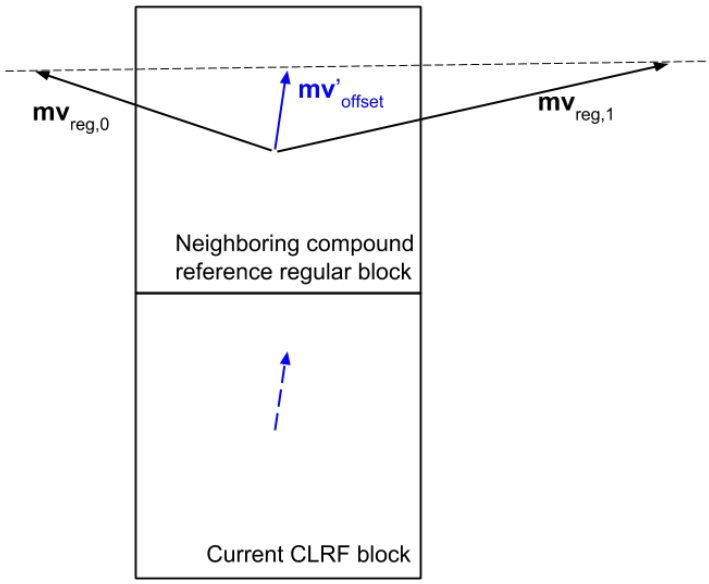


Figure 3.4: Predicting an offset motion vector from regular motion vectors.

Similarly, there are also situations where we try to predict an offset motion vector from regular vectors. For such situations, we incorporate the same idea, but in a reversed manner. As illustrated in figure 3.4, the motion offset prediction is derived by calculating the linearly weighted average of the regular motion vectors $\mathbf{mv}_{reg,0}$ and $\mathbf{mv}_{reg,1}$:

$$\mathbf{mv}'_{offset} = (1 - t_d)\mathbf{mv}_{reg,0} + t_d\mathbf{mv}_{reg,1}. \tag{3.13}$$

By the above approach, we are able to perform motion vector prediction for our proposed scheme with CLRF, which significantly reduces the motion vector

coding redundancy.

3.3.5 Complexity Analysis and Speed Optimization with the Block-Constrained Algorithm

From (3.10), given N pixels, $\mathbf{x}$ is a vector with $2N$ variables. Therefore minimizing the cost J in (3.9) involves solving $2N$ linear equations. Directly solving the inverse of a $2N \times 2N$ matrix A would result in $O(N^3)$ complexity with simple approaches such as the Gaussian elimination method.

However, since the accuracy requirement of our calculation is not quite strict, iterative solutions such as the conjugate gradient (CG) method can be naturally more efficient. Generally, the complexity of CG is $O(N^2M)$, where M is the number of iterations and can be chosen as a much smaller number than $2N$ for our precision requirement.

Furthermore, note A is a sparse matrix with $O(N)$ number of non-zero elements. This is mainly because when applying the Laplacian filter mask, for every pixel, only a constant number of its neighbors (define by the mask) can affect the variables (u, v) associated with the current pixel. Further proof of sparsity of A is not included in this dissertation, but should be fairly straight-forward by writing out D^TD and L^TL in (3.7) and (3.8).

Therefore, considering that at each iteration for CG, the dominant complexity

cost lies in calculating the product of A and a $2N \times 1$ vector, and that such multiplication now takes only $O(N)$ complexity due to the sparsity, the total complexity of optimizing J becomes $O(NM)$.

Therefore, the complexity of optical flow estimation depends on the number of pixels N, as well as number of iterations M. It should also be noted that for larger N, it also takes more iterations to converge to a certain precision, thus M should also increase as we increase N [1].

In addition, as discussed in section 3.3.2, the number of pyramid levels num_P and the number of warping steps num_W at each pyramid level are also factors influencing the overall complexity. Use of median filter to the motion field, which improves the optical flow estimation accuracy, also involves more complexity.

In this section, we present an alternative block-constrained algorithm to lower the complexity of the proposed scheme.

The basic idea of the block-constrained algorithm is simple: instead of performing optical flow estimation for the whole frame, we first divide the current frame f_n into blocks of size $h \times w$, and then perform optical flow estimation for each block **independently**.

At the encoder end, before encoding the current frame, we calculate the optical flow for each such block, and combine the optical flow of each block together to

[1] In fact, it is proven that when $M = 2N$, CG is guaranteed to converge to the exact solution assuming no precision loss. However in our application, such high precision is not required and we still choose $M \ll N$.

form a frame-level motion field. Then, CLRF interpolation and further encoding scheme of the current frame stays the same as in section 3.3.2.

At the decoder, however, the optical flow estimation is not done before decoding the current frame. Instead, as we decode and reconstruct some coding block, only if it uses the CLRF as reference frame for motion compensation, will we perform optical flow estimation for the $h \times w$ blocks in f_{CLRF} that are needed by the current coding block. These $h \times w$ blocks are determined by the location of the current block, the offset motion vector, and the length of sub-pixel interpolation filter L (as illustrated in figure 3.5). If a certain $h \times w$ block in the CLRF is already interpolated for a previously decoded coding block, then skip its calculation and use the previously interpolated result.

The above block-constrained algorithm greatly improves the speed of the proposed scheme because of the following reasons. First, the complexity of performing optical flow estimation for a block is $O(N_b M_b)$, where N_b is the number of pixels in a block ($N_b = hw$), and M_b is the number of iterations needed. Therefore the total complexity of processing all blocks in the frame is $O(\sum\{N_b M_b\}) = O(\sum\{N_b\} M_b) = O(N M_b)$. As mentioned, since N_b is much smaller than N, the number of iterations (M_b) needed to converge to a certain precision is also much smaller than M, thus reducing the total complexity by a large factor. Second, at the decoder, only a portion of $h \times w$ blocks will need optical flow estimation, since there may be many other coding blocks predicting from other references rather

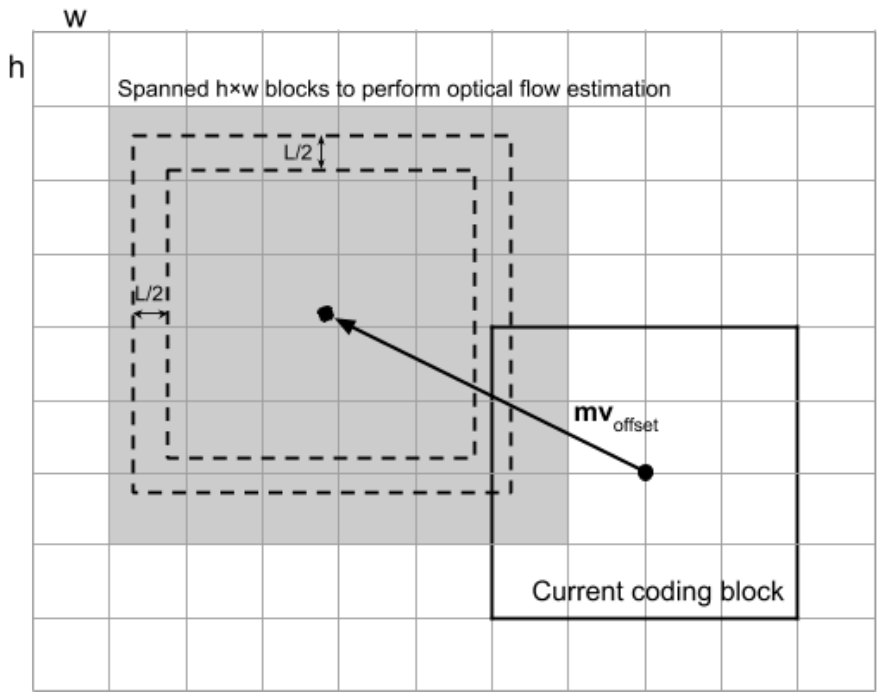


Figure 3.5: Illustration of the block-based algorithm at the decoder. For the current coding block, the $h \times w$ blocks that requires optical flow estimation are marked by the gray blocks, which are determined by the offset motion vector $\mathbf{mv}_{offset}$ and the length of sub-pixel interpolation filter L.

than the CLRF. The decoder complexity is further reduced in this way. Lastly, the optical flow estimation of each block should not interfere with each other, which ensures simple parallel design of the algorithm and can be quite helpful for the hardware design.

The key aspect of the block-based algorithm lies in the fact that the optical flow estimation of each block does not rely on other blocks. This enables the decoder to selectively skip certain blocks according to the usage of CLRF. To ensure such independence, the estimated optical flow of neighboring blocks should not affect the current block through the spatial constraint term. Therefore, we treat

the initialization of the neighboring blocks' motion field as their actual motion. Since the initialization is already available to the decoder before the optical flow estimation, it is considered as part of the constant term c_{spat} in (3.8), effectively relaxing the spatial constraint across the block boundary compared to the frame-based algorithm. Moreover, noting the initialization is not accurate, we apply a lower confidence level to the initialization to decrease its influence on the current block.

Apart from the spatial term, for the data term, the calculation of derivatives also depend on neighboring blocks, since the derivative filter may span out of block. Similarly, the initialization is also used here to generate the derivatives near the block boundaries.

It should be noted that the block-constrained algorithm depends more on the quality of motion field initialization. On the one hand, as we discussed, this is because the initialization is used to handle the block boundaries. On the other hand, the block-constrained algorithm by its nature works only at local areas, thus requiring the initialization to match the current block to the same local area for a better accuracy.

3.3.6 Experimental Results

The proposed scheme with the various design optimizations is integrated with the AV1 framework according to section 3.2.1. Various video sequences were tested

with resolutions including low-res (240p, CIF, etc.), mid-res (480p, 4CIF) and hd-res (720p, 1080p. etc.). In this section, we will discuss experimental results from two aspects: the peak performance and the trade-off between complexity and performance.

For peak performance, frame-based optical flow is utilized with a three-level pyramid structure ($num_P = 3$). The number of warping steps is also set to 3 ($num_W = 3$). Each video sequence is encoded with 150 frames at various target bit-rates (resulting in quality ranging between 35 and 50 dB approximately).

The bit-rate reduction compared to baseline AV1 encoder is shown in table 3.1. It is evident that substantial coding gains are obtained and that the coding gain is consistent across the extensive set of testing video clips with various resolutions. The overall bit-rate reduction is 3%, 3.8% and 3.5% for low-res, mid-res and hd-res respectively. Especially, note the relatively larger gains (approximately 8%-10%) for sequences with very complicated moving subjects (such as crowd_run, rush_field_cuts, ice, etc.) and sequences with non-translational motions (such as station2, blue_sky, city, etc.). This further proves that our proposed algorithm utilizes the motion information more efficiently, and confirms the capability of the estimated per-pixel optical flow.

Recognizing the practical constraints for video coding applications, the additional complexity of performing optical flow estimation (especially at the decoder) is also an important factor. As discussed in section 3.3.5, various parameters could

Table 3.1: Peak performance BD-rate reduction (%) of the proposed method

Low-res		Mid-res		HD-res	
akiyo	-2.97	BQMall	-4.45	basketballdrive	-3.02
basketballpass	-6.32	BasketballDrillText	-3.45	blue_sky	-8.41
blowingbubbles	-2.55	BasketballDrill	-4.21	bqterrace	-1.34
bowing	-3.78	Flowervase	-2.52	cactus	-6.17
bqsquare	-2.50	Keiba	-0.57	chinaspeed	-1.19
bridge_close	-0.57	Mobisode2	-2.79	city	-3.25
bridge_far	0.13	PartyScene	-2.65	crew	-3.80
bus	-2.69	RaceHorses	-4.26	crowd_run	-9.76
cheer	-2.59	aspen	-4.11	cyclists	-3.35
city	-5.67	city	-8.38	dinner	-3.62
coastguard	-1.04	controlled_burn	-1.67	ducks_take_off	-0.71
container	-2.01	crew	-5.38	factory	-4.16
crew	-3.93	crowd_run	-8.98	fourpeople	-3.93
deadline	-3.89	ducks_take_off	-1.06	in_to_tree	-3.89
flower	-2.36	harbour	-2.75	jets	-7.25
flowervase	-2.94	ice	-7.26	johnny	-2.15
football	-2.59	into_tree	-4.94	kimono1	-5.13
foreman	-4.80	old_town_cross	-6.73	kristenandsara	-3.67
garden	-2.27	park_joy	-3.13	life	-4.96
hallmonitor	-1.45	red_kayak	-0.18	mobcal	-0.65
harbour	-1.56	rush_field_cuts	-8.70	night	-4.36
highway	-1.92	sintel_trailer_2k	-1.22	old_town_cross	-4.21
husky	-1.63	snow_mnt	-0.10	parkjoy	-3.64
ice	-7.75	soccer	-3.05	parkrun	-1.24
keiba	-0.51	speed_bag	-1.14	parkscene	-5.20
mobile	-2.88	station2	-8.01	ped	-1.45
mobisode2	-2.33	tears_of_steel1	-2.84	riverbed	-0.22
motherdaughter	-5.76	tears_of_steel2	-5.13	rush_hour	-3.23
news	-3.05	touchdown_pass	-4.17	sheriff	-1.27
pamphlet	-3.40	west_wind_easy	-0.55	shields	-1.86
paris	-5.29			station2	-6.32
racehorses	-6.01			stockholm_ter	-2.40
signirene	-3.01			sunflower	0.51
silent	-3.10			tennis	-0.84
soccer	-2.42			tractor	-3.11
stefan	-1.70			vidyo1	-3.92
students	-4.40			vidyo3	-4.64
tempete	-1.18			vidyo4	-4.55
tennis	-2.23				
waterfall	-3.13				
Average	**-3.00**	**Average**	**-3.81**	**Average**	**-3.48**

influence the complexity of the proposed approach. First, as presented in 3.3.5, the additional complexity should be linear to the the number of iterations for CG, M and M_b. This is confirmed by figure 3.6, which clearly demonstrate such linear relationship for sequence *city_cif*. Note that, as also shown in the figure, even with the same number of iterations ($M = M_b$), the block-constrained algorithm yields a low complexity. This is because for the block-constrained algorithm, the relaxed condition at block boundaries effectively lowers the number of total non-zero elements in the sparse matrix by a constant factor. Furthermore, the decode is able to skip blocks that is not referred to, which also lowers the complexity. Moreover, it should be noted that as the number of iterations increases, the complexity for the block-constrained algorithm eventually becomes a bit lower than the linear trend. This is due to the fact that we terminate the algorithm when a certain precision is reached, and that the block-constrained algorithm with fewer variables converges much faster. In our experiment, we notice that such relationship and similar trend also exists for other tested sequences, and therefore for the rest of the section, we use the complexity of *city_cif* as a coarse approximation of the average complexity.

Next in figure 3.7, the relationship between the coding performance gain for the low-res test set and the additional decoder complexity is presented (the number of iterations, M or M_b, is shown as the label of each data point). For both frame-based and block-constrained algorithm, the performance improves with increasing

Table 3.2: Table of BD-rate reduction (%) with various complexity trade-offs

set	num_P	num_W	method	lowres	midres	HDres	relative complexity
1 (peak)	3	3	Frame	-3.001	-3.813	-3.483	1
2	3	3	Block	-2.293	-3.056	-2.916	0.026
3	1	1	Frame	-2.274	-2.955	-2.883	0.134
4	1	1	Block	-1.879	-2.663	-2.647	0.010
init only	0	0	-	-1.227	-2.076	-2.238	0

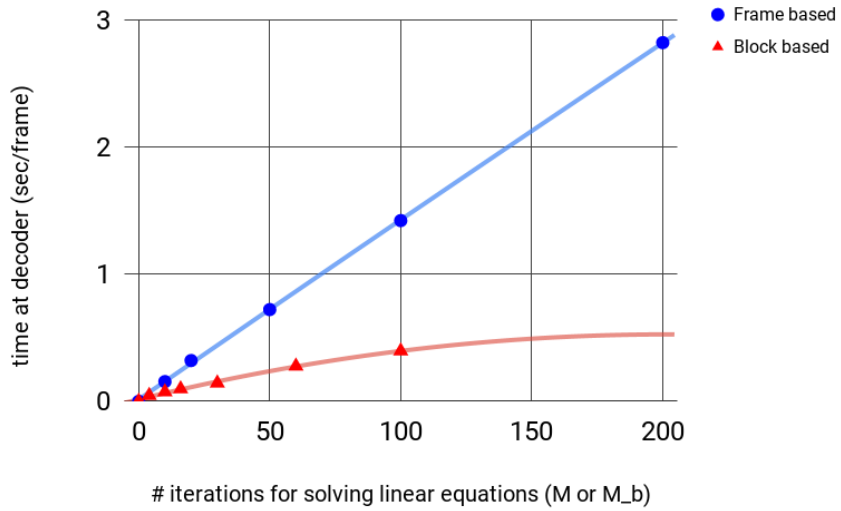


Figure 3.6: The linear relationship of decoder side complexity v.s. number of iterations for solving linear equations (M or M_b).

complexity until it reaches a certain level. Also note that the block-constrained algorithm reaches such level much faster than the frame-based algorithm, but its max performance gain is lower, due to its limit of ignoring the correlation across block boundaries. It can be concluded that, when the desired decoder complexity is limited, the block-constrained algorithm serves as a good alternative in terms of the trade-off between complexity and overall performance.

Such trade-off is also influenced by other parameters, here in table 3.2, we

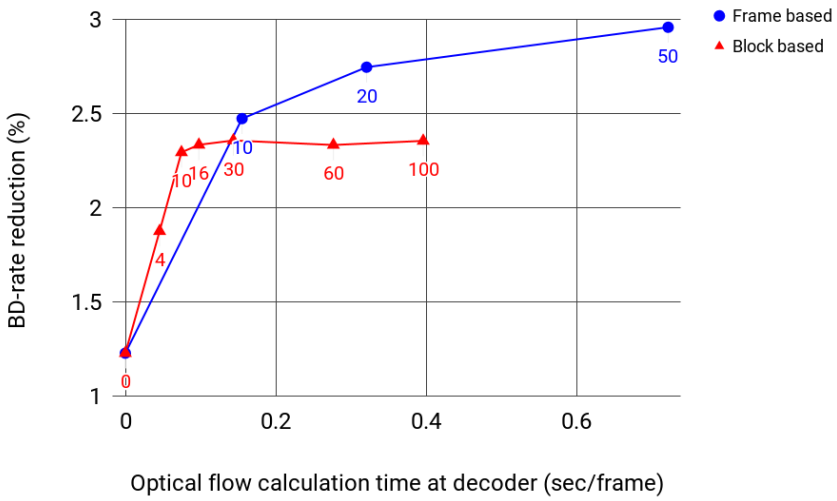


Figure 3.7: Trade-off between performance and complexity for both frame-based and block-constrained algorithms. The label associated with each data point represents the number of iterations M or M_b.

present the average coding gain for a few experiment sets, each with difference choices of num_P, num_W and use of the block-constrained algorithm. Obviously, by changing the parameters, different trade-offs of performance and complexity can be achieved. With the fastest setting (set 4), more than 60% of the coding gains are maintained, while the additional complexity is only 1% as compared to the peak performance.

It is also worth noting that, for the "init only" set, we do not perform optical flow estimation at all, and use the initialization of motion field directly to generate the CLRF (hence yielding nearly no additional complexity). As shown, it also provides a large bit-rate reduction. This clearly shows the effectiveness of our motion vector initialization scheme, and how it lays a great foundation for the

following optical flow estimation.

3.4 Co-located Reference Frame Using the Optimal Linear Estimator

The CLRF generation method in section 3.3 based on per-pixel motion field estimation potentially suffers from quantization errors in the reconstructed reference frames as well as drastically increased complexity due to extensive motion search. Furthermore, some video contents (such as the ones containing object occlusion, gradual change of brightness, etc.) violates the brightness constancy assumption for optical flow estimation and thus resulting in distorted motion field and sub-optimal performance.

In this section, we propose a novel bi-directional motion compensation mode that efficiently utilizes the motion information that is already available to the decoder, without recourse to extensive search. An estimation theory based approach is proposed and utilized to provide a high quality prediction, which adaptively combines contributions from multiple motion-compensated references.

3.4.1 Problem Setup

To interpolate each pixel in the CLRF, the first step of the proposed scheme is to generate candidate motion vectors for this pixel location. Similar to the

method introduced in section 3.3.3, we utilize the previously transmitted and decoded MVs, and project them linearly between the reference frames f_{n_0} and f_{n_1}. However, instead of using the one that intersects the current frame by the closes pixel location to the current pixel, we collect M candidate MVs ($\mathbf{mv}_i$, $i = 1, 2, .., M$), whose intersection pixel locations are closest to the current pixel, to form a candidate MV set.

Given the candidate motion vectors, M references can be generated by linearly combining each reference pair of pixels from the respective frames f_{n_0} and f_{n_1} (denoted as $y_i^{(0)}$ and $y_i^{(1)}$). The combined references are denoted $\mathbf{x} = (x_1, x_2, ..., x_M)^T$. For now let us assume the simple case where $n - n_0 = n_1 - n$ to illustrate the basic idea (then $x_i = 0.5y_i^{(0)} + 0.5y_i^{(1)}$), noting that derivation for a general setup is straightforward.

From an estimation theory perspective, we regard the references x_i as M observations correlated with y. A linear estimator with weight vector $\mathbf{w} = (w_1, w_2, ..., w_M)^T$ is used to generate an estimate of y:

$$\tilde{y} = \mathbf{w}^T \mathbf{x}. \tag{3.14}$$

The weights $\mathbf{w}$ are determined such that they minimize the mean squared

prediction error:

$$\mathbf{w} = argmin_{\mathbf{w}}\{E\{(y - \tilde{y})^2\}\}. \tag{3.15}$$

A standard result in linear estimation theory, obtained by simple calculus, converts the problem into a set of linear equations:

$$E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w} - E\{\mathbf{x}y\} = 0, \tag{3.16}$$

where $E\{\mathbf{x}\mathbf{x}^T\}$ is the $M \times M$ correlation matrix of $\mathbf{x}$, and $E\{\mathbf{x}y\}$ is the $M \times 1$ vector containing the cross correlations of x_i and y.

However, how to obtain the proper correlation matrix and the cross correlation vector still remains a challenging problem. As presented in [58] and [59], offline training of the weights can be performed by collecting relevant data from the video codecs to provide estimates of the relevant correlations. In section 3.4.2, we propose a different online adaptive method that utilizes the bi-directional prediction scheme to estimate such correlations on-the-fly to better adapt to local statistics.

3.4.2 CLRF Interpolation Based on Optimal Design of Adaptive Linear Estimators

First, let us consider the actual motion trajectory of the pixel of interest. As shown in Figure 3.8, the motion trajectory is represented by the dashed line, and the pixel values of the pixels at frame f_{n_0} and f_{n_1} are denoted by $y^{(0)}$ and $y^{(1)}$. Let us further assume an auto-regressive (AR) model along the motion trajectory. Since $n - n_0 = n_1 - n$, the correlations are symmetric, and we have:

$$\begin{aligned} y &= \rho_t y^{(0)} + v, \\ y^{(1)} &= \rho_t y + v^{(1)}, \end{aligned} \tag{3.17}$$

where ρ_t is the temporal correlation coefficient. Assuming constant variance σ_y^2, we have: $\rho_t = \frac{E(y^{(0)}y)}{\sigma_y^2} = \frac{E(yy^{(1)})}{\sigma_y^2}$. v and $v^{(1)}$ are white noise random variables that are uncorrelated with y, $y^{(0)}$ and $y^{(1)}$.

Given the AR model of (3.17), the correlation coefficient between $y^{(0)}$ and $y^{(1)}$, denoted ρ_{12}, is:

$$\rho_{12} = \frac{E(y^{(0)}y^{(1)})}{\sigma_y^2} = \rho_t^2. \tag{3.18}$$

Now consider a motion vector candidate $\mathbf{mv}_i$, which in general may not be the same as the true motion trajectory as illustrated in figure 3.8. Here, the separable spatial-temporal correlation model is further assumed, such that the correlation

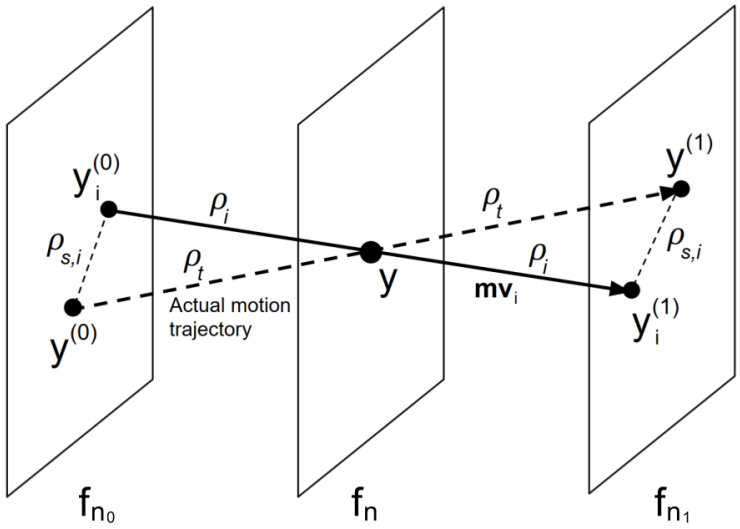


Figure 3.8: Illustration of cross correlation calculation. Along the real motion trajectory (the dashed line), ρ_t denotes the temporal correlation coefficient. For each of the MV candidates $\mathbf{mv}_i$, the cross correlation coefficient ρ_i is given by the separable model: $\rho_i = \rho_t \rho_{s,i}$, where $\rho_{s,i}$ is the spatial correlation coefficient.

coefficient, $\rho_i = \frac{E\{y_i^{(0)} y\}}{\sigma_y^2} = \frac{E\{y y_i^{(1)}\}}{\sigma_y^2}$ is given by:

$$\rho_i = \rho_t \rho_{s,i}, \tag{3.19}$$

where $\rho_{s,i}$ is the spatial correlation coefficient relevant to the distance between $y^{(0)}$ and $y_i^{(0)}$ (which equals the distance between $y^{(1)}$ and $y_i^{(1)}$ in this setting due to symmetry).

Analogous to (3.18), the correlation coefficient between $y_i^{(0)}$ and $y_i^{(1)}$, denoted $\rho_{12,i}$ can be related to ρ_i:

$$\rho_{12,i} = \rho_i^2. \tag{3.20}$$

Now, for a given $\mathbf{mv}_i$, the reference pixel value is $x_i = 0.5 y_i^{(0)} + 0.5 y_i^{(1)}$. With

the same pixel value variance $\sigma_x = \sigma_y = \sigma$, we obtain:

$$E\{x_i y\} = \rho_i \sigma^2 = \sqrt{\rho_{12,i}}\sigma^2. \tag{3.21}$$

Since the two reference frames are already reconstructed, we propose to estimate $\rho_{12,i}$ on the fly by collecting neighboring data (a 5×5 patch) around pixels $y_i^{(0)}$ and $y_i^{(1)}$. Once $\rho_{12,i}$ is calculated for each $i = 1, 2, ..., M$, the cross correlation vector is calculated according to (3.21).

Next, the remaining ingredient needed to determine the linear estimator by solving (3.16), is the correlation matrix, i.e., we need the correlations between candidates $E\{x_{i_1} x_{i_2}\}$ for any i_1 and i_2. From (3.21), for a given i, we can write x_i as:

$$x_i = \rho_i y + z_i, \tag{3.22}$$

where z_i is the "innovation" in x_i, that is, what is uncorrelated with y. Therefore,

$$E\{x_{i_1} x_{i_2}\} = \rho_{i_1}\rho_{i_2}\sigma^2 + E\{z_{i_1} z_{i_2}\}, \tag{3.23}$$

where $E\{z_{i_1} z_{i_2}\}$ is the correlation of the innovations relative to y. We propose to model this correlation with an exponential decay model, i.e., the correlation drops exponentially with the distance from the referred pixels. Note that this distance equals the difference between the two candidate motion vectors, therefore we have:

$$E\{z_{i_1} z_{i_2}\} = exp(-\alpha||\Delta\mathbf{mv}_{i_1,i_2}||)\sigma_{z_1}\sigma_{z_2}, \tag{3.24}$$

where $\Delta\mathbf{mv}_{i_1,i_2} = \mathbf{mv}_{i_1} - \mathbf{mv}_{i_2}$, and the variance of z_i is given by $\sigma_{z_i}^2 = (1-\rho_i^2)\sigma^2$.

Substituting (3.24) into (3.23), we obtain an estimate for the correlation matrix, and (3.21) provides the cross correlation vector. Therefore the corresponding linear estimator weights can now be obtained by (3.16) (note that the variance σ^2 cancels out and thus is not needed). The resulting linear estimator is used to generate the interpolated motion compensated prediction for this pixel location.

We re-emphasize that we assumed $n - n_0 = n_1 - n$ in the above derivations for simplicity of presentation. It is straightforward to derive the results for other settings using the same logic, and the details are omitted here for conciseness. Also, the derivation assumed zero-mean random variables, which can be approximated by subtracting a local mean or a constant bias.

3.4.3 MV Prediction Scheme Based on Correlation Model Consistency

Conventionally, the motion vectors are predicted by another neighboring motion vector, or by a projected motion vector from other existing motion vectors. Inspired by the method introduced in section 3.4, in this section, we proposed an innovative approach, which considers multiple existing MVs as observations of

the current MV and generates an MV prediction based on such observations and statistical models of the pixel values associated with such MV observations.

Similar to the method in section 3.4, first, a set of MV candidates, denoted as $\mathbf{mv}_i$, are generated. Note the motion vector candidates are all projected such that they points from reference frame f_{n_0} to f_{n_1}, intersecting the current frame. As shown in figure 3.9, such MV candidates are treated as observations of the actual motion vector ($\mathbf{mv}_t$, represented by dash line), which is unknown and to be predicted.

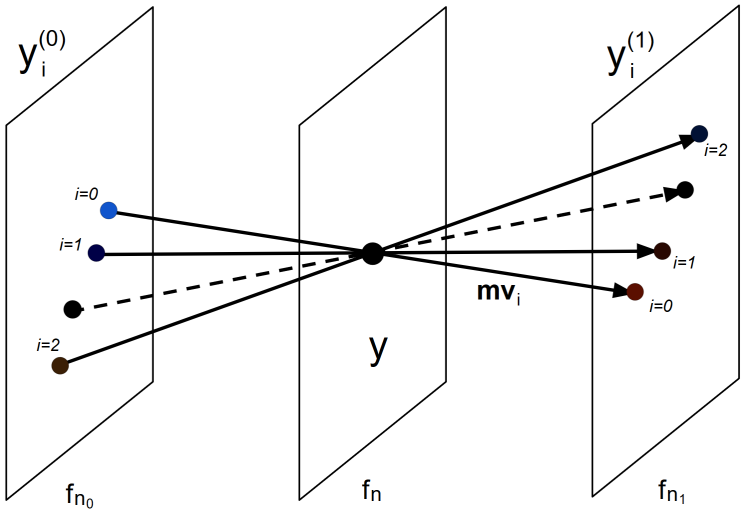


Figure 3.9: Illustration of motion vector candidates.

Now let us consider the reconstructed pixels associated with each MV candidate $\mathbf{mv}_i$, denoted as $y_i^{(0)}$ and $y_i^{(1)}$. As shown in (3.20), assuming that the pixels form an AR process and that the pixel correlation follows the separated correlation model, the correlation coefficient between them, ρ_i, is given by the multiplication

of the temporal correlation coefficient ρ_t and the spatial correlation coefficient $\rho_{s,i}$.

Note that the temporal correlation captures the correlation along the motion trajectory, and thus ρ_t should also be the temporal correlation coefficient between the pixels associated with $\mathbf{mv}_t$. Assuming stationary statistics of pixels, such ρ_t can be estimated from previous decoded frames and MVs.

Moreover, with the exponential decay model of the spatial correlation, similar to (3.24), $\rho_{s,i}$ depends on the spatial distance of the associated pixel location of $\mathbf{mv}_i$ to that of $\mathbf{mv}_t$, which can be calculated as:

$$\rho_{s,i} = exp(-\beta||\mathbf{mv}_i - \mathbf{mv}_t||_2), \tag{3.25}$$

where β is a parameter controlling the speed of decay that can also be estimated from previously reconstructed pixels.

Then, on the one hand, with the estimated ρ_t and β available, it can be concluded from (3.20) and (3.25) that ρ_i is a function of the actual motion vector of the current pixel, denoted as $\rho_i(\mathbf{mv}_t)$.

On the other hand, it is possible to estimate ρ_i by collecting neighboring pixels around $y_i^{(0)}$ and $y_i^{(1)}$ and treating the neighboring pixels as realizations of the associated pixel. These estimated values of correlation coefficients associated with each candidate motion vector is denoted as $\bar{\rho}_i$

Therefore we propose to generate the prediction the actual motion vector,

denoted as $\tilde{\mathbf{mv}}_t$ of the pixel by the following cost minimization formula:

$$\tilde{\mathbf{mv}}_t = \underset{\mathbf{mv}_t}{\operatorname{argmin}} \sum_i ||\rho_i(\mathbf{mv}_t) - \bar{\rho}_i||. \tag{3.26}$$

It is not straight-forward to derive a closed-form solution for (3.26). Instead, we propose to perform a "greedy search" of $\mathbf{mv}_t$ using existing motion search methods to provide a good prediction. It is very important to stress here that we only utilize the search patterns of such motion estimation methods, but do not suffer from their dramatic complexity cost, due to the fact that for every search iteration, it is not necessary for us to compare pixel values of the neighboring block or to perform sub-pixel interpolation filters. The only calculation needed for each iteration is merely calculate $\rho_i(\mathbf{mv}_t)$, which involves minimal complexity.

3.4.4 Experimental Results

The proposed method was implemented in the AV1 video coding software [4]. Two sets of experiments were conducted, where the *baseline* set utilizes the regular compound mode in the AV1 codec, while the *proposed* set adds one extra prediction mode that predicts from the co-located reference frame interpolated by our proposed method. The performance was evaluated over a diverse set of sequences that vary in resolution and motion characteristics, and results are provided for a wide range of bit rates. For each sequence, 100 frames are encoded. Also, in our

Table 3.3: BD-rate reduction using the proposed method.

Sequence	BD-rate change (%)
coastguard	-1.53
mobile	-4.37
foreman	-7.17
flower	-3.19
bus	-2.96
BasketballPass	-4.81
container	-3.95
tempete	-1.30
Average	**-3.75**

experiments, we set the parameters $M = 9$ and $\alpha = 0.1$.

The BD-rate reduction [16] of the proposed method compared to the baseline is shown in Table 3.3. It can be clearly seen that significant improvement (**3.75%** BD-rate reduction on average) is obtained with the proposed method. As an example, the rate-distortion (R-D) curve of bus_cif.yuv (which contains dramatic motions across the sequence) is shown in Figure 3.10. As can be seen, consistent gain is achieved for a wide range of bit-rate, which further proves the effectiveness of the method.

It should be emphasized that our method not only brings significant bit-rate reduction, but does so at minimal complexity cost to the decoder, since we do not require it to perform motion search, but use previously decoded information instead.

Moreover, while the effectiveness of the proposed method was evidenced by the above AV1 results, it must be stressed that the design principles are generally

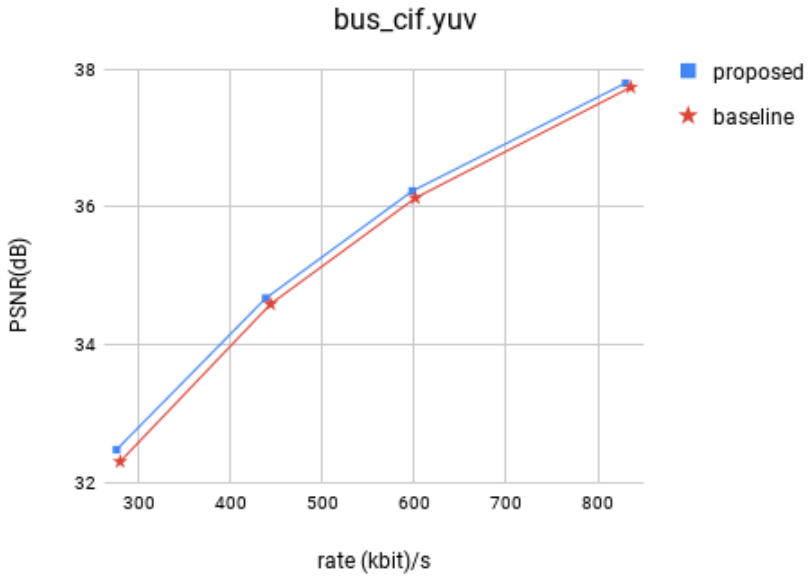


Figure 3.10: The R-D curve for bus_cif.yuv.

applicable to other video codecs that employ motion-compensated prediction with hierarchical structures, such as H.264, HEVC, etc.

3.5 Conclusion

This chapter focuses on the bi-directional motion compensated predictive coding scheme with CLRF.

The CLRF scheme is first introduced to account for available motion information to the decoder, while providing a practical yet effective method to address the problem of possible offset from the assumed linear motions.

To generate the CLRF, we first propose an approach based on optical flow

estimation, which extracts per-pixel motion from the available reconstructed reference frames. Various optimization techniques, including motion field initialization, motion vector prediction with the offset motion vector are also presented. Recognizing the increased complexity, a block-constrained fast algorithm is also introduced, which requires much less calculation with an acceptable trade-off of performance.

Based on the optimal estimation principles, we further proposed another approach which generates the CLRF using optimal linear estimators. Without extensive motion estimation, existing transmitted motion vectors are re-used, where their associated reference blocks are treated as observations of the predicted blocks. A method of estimating the cross-correlation matrix as well as the correlation vector is derived, which enables the encoder to adaptively design optimal linear estimators on-the-fly. Moreover, inspired by such approach, a novel approach of motion vector prediction is also proposed, where the prediction is given by the motion vector that is most consistent with the observed statistics.

As illustrated by the experimental results, both methods brings significant bit-rate reduction and the effectiveness of such bi-directional motion compensated prediction scheme is proved.

Bibliography

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, *Overview of the h. 264/avc video coding standard*, *IEEE Trans. Circ. Sys. Video Tech.* **13** (Jul, 2003) 560–576.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, *Overview of the high efficiency video coding (hevc) standard*, *IEEE Trans. Circ. Sys. Video Tech.* **22** (Dec, 2012) 1649–1668.

[3] 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 *Proc. SPIE 8666, Visual Information Processing and Communication IV, 866606*, Feb, 2013.

[4] Y. Chen et al., *An overview of core coding tools in the av1 video codec*, in *Picture Coding Symposium (PCS)*, pp. 24–27, 2018.

[5] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, *Error resilient video coding techniques*, *IEEE Sig. Proc. Mag.* **17** (Jul, 2000) 61–82.

[6] R. Zhang, S. L. Regunathan, and K. Rose, *Video coding with optimal inter/intra-mode switching for packet loss resilience*, *IEEE Jrnl. Sel. Areas Comm.* **18** (Jun, 2000) 966–976.

[7] H. Yang and K. Rose, *Advances in recursive per-pixel end-to-end distortion estimation for robust video coding in h. 264/avc*, *IEEE Trans. Circ. Sys. Video Tech.* **17** (Jul, 2007) 845–856.

[8] J. Han, V. Melkote, and K. Rose, *Transform-domain temporal prediction in video coding: exploiting correlation variation across coefficients*, in *IEEE ICIP*, Sep, 2010.

[9] J. Han, V. Melkote, and K. Rose, *A recursive optimal spectral estimate of end-to-end distortion in video communications*, in *Proc. Packet Video*, Dec, 2010.

[10] B. A. Heng, J. G. Apostolopoulos, and J. S. Lim, *End-to-End Rate-Distortion Optimized MD Mode Selection for Multiple Description Video Coding*, *EURASIP Jrnl. App. Sig. Proc.* (2006).

[11] Y. Liao and J. D. Gibson, *Rate-distortion based mode selection for video coding over wireless networkswith burst losses*, in *17th International Packet Video Workshop*, May, 2009.

[12] K. Rose and S. L. Regunathan, *Toward optimality in scalable predictive coding*, *IEEE Trans. Img. Proc.* **10** (Jul, 2001) 965–976.

[13] J. Han, V. Melkote, and K. Rose, *Estimation-theoretic delayed decoding of predictively encoded video sequences*, in *Proc. IEEE DCC*, Mar, 2010.

[14] S. Li, T. Nanjundaswamy, Y. Chen, and K. Rose, *Asymptotic closed-loop design for transform domain temporal prediction*, in *IEEE ICIP*, Sep, 2015.

[15] J. Han, V. Melkote, and K. Rose, *A spectral approach to recursive end-to-end distortion estimation for sub-pixel motion-compensated video coding*, in *IEEE ICASSP*, May, 2011.

[16] G. Bjontegaard, *Calcuation of average psnr differences between rd-curves*, *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4* (Apr, 2001).

[17] J. Xin, K. N. Ngan, and G. Zhu, *Combined inter-intra prediction for high definition video coding*, in *Picture Coding Symposium*, 2007.

[18] Y. Chen, K. Rose, J. Han, and D. Mukherjee, *A pre-filtering approach to exploit decoupled prediction and transform block structures in video coding*, in *IEEE ICIP*, Oct, 2014.

[19] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, *A survey of recent results in networked control systems*, *Proceedings of the IEEE* **95** (Jan, 2007) 138–162.

[20] J. H. Braslavsky, R. H. Middleton, and J. S. Freudenberg, *Feedback stabilization over signal-to-noise ratio constrained channels*, *IEEE Transactions on Automatic Control* **52** (Aug, 2007) 1391–1403.

[21] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, *Kalman filtering with intermittent observations*, *IEEE Transactions on Automatic Control* **49** (Sept, 2004) 1453–1464.

[22] S. L. Howard, C. Schlegel, and K. Iniewski, *Error control coding in low-power wireless sensor networks: When is ecc energy-efficient?*, *EURASIP Journal on Wireless Communications and Networking* **2006** (2006), no. 2 29–29.

[23] M. Fu and C. E. de Souza, *State estimation for linear discrete-time systems using quantized measurements*, *Automatica* **45** (2009), no. 12 2937–2945.

[24] W. S. Wong and R. W. Brockett, *Systems with finite communication bandwidth constraints. II. stabilization with limited information feedback*, *IEEE Transactions on Automatic Control* **44** (May, 1999) 1049–1053.

[25] S. Tatikonda and S. Mitter, *Control under communication constraints*, *IEEE Transactions on Automatic Control* **49** (July, 2004) 1056–1068.

[26] A. Ribeiro, G. B. Giannakis, and S. I. Roumeliotis, *Soi-kf: Distributed kalman filtering with low-cost communications using the sign of innovations*, *IEEE Transactions on Signal Processing* **54** (Dec, 2006) 4782–4795.

[27] K. You, L. Xie, S. Sun, and W. Xiao, *Multiple-level quantized innovation kalman filter*, *IFAC Proceedings Volumes* **41** (2008), no. 2 1420–1425.

[28] S. Dey, A. Chiuso, and L. Schenato, *Remote estimation with noisy measurements subject to packet loss and quantization noise*, *IEEE Transactions on Control of Network Systems* **1** (2014), no. 3 204–217.

[29] M. Nourian, A. S. Leong, S. Dey, and D. E. Quevedo, *An optimal transmission strategy for kalman filtering over packet dropping links with imperfect acknowledgements*, *IEEE Transactions on Control of Network Systems* **1** (2014), no. 3 259–271.

[30] Y. Boers and H. Driessen, *Modified riccati equation and its application to target tracking*, *IEEE Proceedings - Radar, Sonar and Navigation* **153** (Feb, 2006) 7–12.

[31] L.-K. Liu and E. Feig, *A block-based gradient descent search algorithm for block motion estimation in video coding*, *IEEE Transactions on Circuits and Systems for Video Technology* **6** (1996), no. 4 419–422.

[32] S. Zhu and K.-K. Ma, *A new diamond search algorithm for fast block-matching motion estimation*, *IEEE Transactions on Image Processing* **9** (2000), no. 2 287–290.

[33] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, and L.-G. Chen, *Survey on block matching motion estimation algorithms and architectures with new results*, *Journal of VLSI signal processing systems for signal, image and video technology* **42** (2006), no. 3 297–320.

[34] 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.

[35] B. K. Horn and B. G. Schunck, *Determining optical flow*, *Artificial intelligence* **17** (1981), no. 1-3 185–203.

[36] D. Sun, S. Roth, and M. J. Black, *A quantitative analysis of current practices in optical flow estimation and the principles behind them*, *International Journal of Computer Vision* **106** (2014), no. 2 115–137.

[37] T. Brox and J. Malik, *Large displacement optical flow: descriptor matching in variational motion estimation*, *IEEE transactions on pattern analysis and machine intelligence* **33** (2011), no. 3 500–513.

[38] L. Xu, J. Jia, and Y. Matsushita, *Motion detail preserving optical flow estimation*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **34** (2012), no. 9 1744–1757.

[39] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, *Bilateral filtering-based optical flow estimation with occlusion detection*, in *European conference on computer vision*, pp. 211–224, Springer, 2006.

[40] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, *Deepflow: Large displacement optical flow with deep matching*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1385–1392, 2013.

[41] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, *Flownet: Learning optical flow with convolutional networks*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2758–2766, 2015.

[42] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, *Flownet 2.0: Evolution of optical flow estimation with deep networks*, in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 2, p. 6, 2017.

[43] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, *Unsupervised deep learning for optical flow estimation.*, in *AAAI*, vol. 3, p. 7, 2017.

[44] F. Kendoul, I. Fantoni, and K. Nonami, *Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles*, *Robotics and Autonomous Systems* **57** (2009), no. 6 591–602.

[45] A. G. Bors and I. Pitas, *Optical flow estimation and moving object segmentation based on median radial basis function network*, *IEEE Transactions on Image Processing* **7** (1998), no. 5 693–702.

[46] R. Krishnamurthy, J. W. Woods, and P. Moulin, *Frame interpolation and bidirectional prediction of video using compactly encoded optical-flow fields and label fields*, *IEEE transactions on circuits and systems for video technology* **9** (1999), no. 5 713–726.

[47] R. Krishnamurthy, P. Moulin, and J. Woods, *Optical flow techniques applied to video coding*, in *International Conference on Image Processing*, vol. 1, pp. 570–573, IEEE, 1995.

[48] S. Lin, Y. Q. Shi, and Y.-Q. Zhang, *An optical flow based motion compensation algorithm for very low bit-rate video coding*, in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2869–2872, IEEE, 1997.

[49] Y. M. Chi, T. D. Tran, and R. Etienne-Cummings, *Optical flow approximation of sub-pixel accurate block matching for video coding*, in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. I–1017, IEEE, 2007.

[50] S. Klomp, M. Munderloh, Y. Vatis, and J. Ostermann, *Decoder-side block motion estimation for h. 264/mpeg-4 avc based video coding*, in *2009 IEEE International Symposium on Circuits and Systems*, pp. 1641–1644, IEEE, 2009.

[51] Y. Chin and C.-J. Tsai, *Dense true motion field compensation for video coding*, in *2013 20th IEEE International Conference on Image Processing (ICIP)*, pp. 1958–1961, IEEE, 2013.

[52] S. Klomp, M. Munderloh, and J. Ostermann, *Decoder-side hierarchical motion estimation for dense vector fields*, in *Picture Coding Symposium (PCS), 2010*, pp. 362–365, IEEE, 2010.

[53] A. Alshin, E. Alshina, and T. Lee, *Bi-directional optical flow for improving motion compensation*, in *28th Picture Coding Symposium*, pp. 422–425, Dec, 2010.

[54] A. Alshin and E. Alshina, *Bi-directional optical flow for future video codec*, in *2016 Data Compression Conference (DCC)*, pp. 83–90, March, 2016.

[55] J. Nocedal and S. J. Wright, *Conjugate gradient methods*, *Numerical optimization* (2006) 101–134.

[56] B. Li, J. Han, and Y. Xu, *Co-located reference frame interpolation using optical flow estimation for video compression*, in *2018 Data Compression Conference*, pp. 13–22, March, 2018.

[57] J. Han, J. Feng, Y. Teng, Y. Xu, and J. Bankoski, *A motion vector entropy coding scheme based on motion field referencing for video compression*, in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 3618–3622, IEEE, 2018.

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

[59] W. Lin, T. Nanjundaswamy, and K. Rose, *Adaptive interpolated motion-compensated prediction with variable block partitioning*, in *Data Compression Conference*, pp. 23–31, 2018.