Prediction of Transmission Distortion for Wireless Video Communication: Algorithm and Application

Zhifeng Chen and Dapeng Wu[1]

Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida 32611

Abstract

In this paper, we develop algorithms for estimating transmission distortion in wireless video communication systems. By leveraging the analytical results obtained in our previous paper (Part I), we design low complexity algorithms that are capable of estimating transmission distortion accurately. We also extend our algorithm for pixel-level transmission distortion estimation to pixel-level end-to-end distortion estimation. Furthermore, we apply our pixel-level end-to-end distortion estimation algorithm to prediction mode decision in H.264 encoder. Experimental results show that 1) our transmission distortion estimation algorithm is more accurate and more robust against inaccurate channel estimation than existing distortion estimation algorithms; 2) our mode decision algorithm achieves remarkable PSNR gain over the existing algorithms for prediction mode decision in H.264 encoder, e.g., an average PSNR gain of 1.44dB for 'foreman' sequence when Packet Error Probability (PEP) equals 5%.

Keywords: Wireless video, transmission distortion, clipping noise, slice data partitioning, unequal error protection (UEP), prediction mode decision.

1. Introduction

Transmitting video over wireless with good quality or low end-to-end distortion is particularly challenging since the received video is subject to not only quantization distortion but also transmission distortion (i.e., video distortion caused by packet errors). The capability of predicting transmission distortion can assist in designing video encoding and transmission schemes that achieve maximum video quality or minimum end-to-end video distortion. In Ref. [1], we have theoretically derived formulae for transmission distortion. In this paper, we leverage the analytical results in Ref. [1] to design algorithms for estimating transmission distortion; we also develop an algorithm for estimating end-to-end distortion, and apply it to prediction mode decision in H.264 encoder.

[1]Please direct all correspondence to Prof. Dapeng Wu, University of Florida, Dept. of Electrical & Computer Engineering, P.O.Box 116130, Gainesville, FL 32611, USA. Tel. (352) 392-4954. Fax (352) 392-0044. Email: wu@ece.ufl.edu. Homepage: http://www.wu.ece.ufl.edu.

To estimate frame-level transmission distortion (FTD), several linear model based algorithms [2, 3, 4, 5] have been proposed. These algorithms use the sum of the newly induced distortion in the current frame and the propagated distortion from previous frames, to estimate transmission distortion. The linear model based algorithms simplify the analysis of transmission distortion at the cost of sacrificing the prediction accuracy by neglecting the correlation between the newly induced error and the propagated error. Liang et al. [6] extend the result in Ref. [2] by addressing the effect of correlation. However, they do not consider the effect of motion vector (MV) error on transmission distortion and their algorithm is not tested with high motion video content. Under this condition, they claim that the LTI models [2, 3] under-estimate transmission distortion due to positive correlation between two adjacent erroneous frames. In Ref. [1], we identify that the MV concealment error is negatively correlated with the propagated error and this correlation dominates over all other types of correlation especially for high motion video. As long as MV transmission errors exist, the transmission distortion estimated by LTI models becomes over-estimated. In Ref. [1], we also quantify the effects of those correlations on transmission distortion by a system parameter called correlation ratio. On the other hand, none of existing works analyzes the impact of clipping noise on transmission distortion. In Ref. [1], we prove that clipping noise reduces the propagated error and quantify its effect by another system parameter called propagation factor. In this paper, we design algorithms to estimate correlation ratio and propagation factor, which facilitates the design of a low complexity algorithm called *RMPC-FTD algorithm* for estimating frame-level transmission distortion. Experimental results demonstrate that our RMPC-FTD algorithm is more accurate and more robust than existing algorithms. Another advantage of our RMPC-FTD algorithm is that all parameters in the formula derived in Ref. [1] can be estimated by using the instantaneous frame statistics and channel conditions, which allows the frame statistics to be time-varying and the error processes to be non-stationary. However, existing algorithms estimate their parameters by using the statistics averaged over multiple frames and assume these statistics do not change over time; their models all assume the error process is stationary. As a result, our RMPC-FTD algorithm is more suitable for real-time video communication.

For pixel-level transmission distortion (PTD), the estimation algorithm is similar to the FTD estimation algorithm since the PTD formula is a special case of the FTD formula as discussed in Ref. [1]. However, in some existing video encoders, e.g., H.264 reference code JM14.0 [7], motion estimation and prediction mode decision are separately considered. Therefore, the MV and corresponding residual are known for distortion estimation in mode decision. In such a case, the PTD estimation algorithm can be simplified with known values of the MV and corresponding residual, compared to using their statistics. In this paper, we design a PTD estimation algorithm, called *RMPC-PTD* for such a case; we also extend RMPC-PTD to estimate pixel-level end-to-end distortion (PEED).

PEED estimation is important for designing optimal encoding and transmission schemes. Some existing PEED estimation algorithms are proposed in Refs. [8, 9]. In Ref. [8], the recursive optimal per-pixel estimate (ROPE) algorithm is proposed to estimate the PEED by recursively calculating the first and second moments of the reconstructed pixel value. However, the ROPE algorithm neglects the significant effect of clipping noise on transmission

distortion, resulting in inaccurate estimate. Furthermore, the ROPE algorithm requires intensive computation of correlation coefficients when pixel averaging operations (e.g., in interpolation filter and deblocking filter) are involved [10], which reduces its applicability in H.264 video encoder. Stockhammer et al. [9] propose a distortion estimation algorithm by simulating K independent decoders at the encoder side during the encoding process and averaging the distortions of these K decoders. This algorithm is based on the Law of Large Number (LLN), i.e., the estimated distortion will asymptotically approach the expected distortion as K goes to infinity. For this reason, we call the algorithm in Ref. [9] as LLN algorithm. However, for LLN algorithm, the larger number of decoders simulated, the higher computational complexity and the larger memory required. As a result, LLN algorithm is not suitable for real-time video communication. To enhance estimation accuracy, reduce complexity and improve extensibility, in this paper, we extend RMPC-PTD algorithm to PEED estimation; the resulting algorithm is called *RMPC-PEED*. Compared to ROPE algorithm, RMPC-PEED algorithm is more accurate since the significant effect of clipping noise on transmission distortion is considered. Another advantage over ROPE algorithm is that RMPC-PEED algorithm is much easier to be extended to support averaging operations, e.g., interpolation filter. Compared to LLN algorithm, the computational complexity and memory requirement of RMPC-PEED algorithm are much lower and the estimated distortion has smaller variance.

In existing video encoders, prediction mode decision is to choose the best prediction mode in the sense of minimizing the Rate-Distortion (R-D) cost for each Macroblock (MB) or sub-MB. Estimation of the MB level or sub-MB level end-to-end distortion for different prediction modes is needed. In inter-prediction, the reference pixels of the same encoding block may belong to different blocks in the reference frame; therefore, PEED estimation is needed for calculating R-D cost in prediction mode decision. In this paper, we apply our RMPC-PEED algorithm to prediction mode decision in H.264; the resulting algorithm is called *RMPC-MS*. Experimental results show that, for prediction mode decision in H.264 encoder, our RMPC-MS algorithm achieves an average PSNR gain of 1.44dB over ROPE algorithm for 'foreman' sequence under $PEP = 5\%$; and it achieves an average PSNR gain of 0.89dB over LLN algorithm for 'foreman' sequence under $PEP = 1\%$.

The rest of paper is organized as follows. Section 2 presents the system description and transmission distortion formulae, which serves as the necessary preliminaries of the RMPC algorithm design. Section 3 presents our algorithms for estimating FTD under two scenarios: one without acknowledgement feedback and one with acknowledgement feedback. In Section 4, we develop algorithms for estimating PTD. In Section 5, we extend our PTD estimation algorithm to PEED estimation. In Section 6, we apply our PEED estimation algorithm to prediction mode decision in H.264 encoder and compare its complexity with existing algorithms. Section 7 shows the experimental results that demonstrate accuracy and robustness of our distortion estimation algorithm and superior R-D performance of our mode decision scheme over existing schemes. Section 8 concludes the paper.

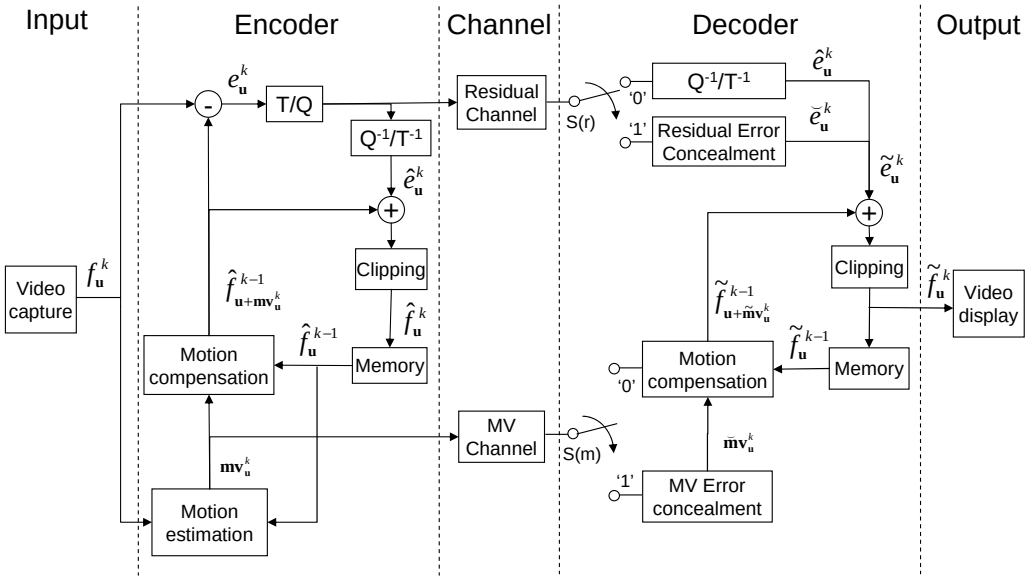


Figure 1: System structure, where T, Q, Q^{-1}, and T^{-1} denote transform, quantization, inverse quantization, and inverse transform, respectively.

2. System Description and Transmission Distortion Formulae

In this subsection, we presents the system description and transmission distortion formulae, which serves as the necessary preliminaries of the RMPC algorithm design. For completeness, the derivation approach and intuitive meaning are also provided in Appendix A.

2.1. Structure of a Wireless Video Communication System

Fig. 1 shows the structure of a typical wireless video communication system. It consists of an encoder, two channels and a decoder where residual packets and MV packets are transmitted over their respective channels. Note that in this system, both residual channel and MV channel are application-layer channels.

2.2. Transmission Distortion Formulae for PTD and FTD

2.2.1. Transmission Distortion Formulae for FTD

Ref. [1] derives a formula for FTD under single-reference prediction, i.e.,

$$D^k = D^k(r) + D^k(m) + D^k(P) + D^k(c), \tag{1}$$

where

$$D^k(r) = E[(\varepsilon^k)^2] \cdot \bar{P}^k(r); \tag{2}$$

$$D^k(m) = E[(\xi^k)^2] \cdot \bar{P}^k(m); \tag{3}$$

$$D^k(P) = \bar{P}^k(r) \cdot D^{k-1} + (1-\beta^k) \cdot (1-\bar{P}^k(r)) \cdot \alpha^k \cdot D^{k-1}; \tag{4}$$

$$D^k(c) = (\lambda^k - 1) \cdot D^k(m); \tag{5}$$

$D^k(r)$ represents the residual packet error induced distortion; $D^k(m)$ represents the MV packet error induced distortion; $D^k(P)$ represents the propagation and clipping noise induced distortion; $D^k(c)$ represents the correlation, among residual error, MV error propagated error and clipping noise, induced distortion; ε^k is the residual concealment error and $\bar{P}^k(r)$ is the weighted average PEP of all residual packets in the k-th frame; ξ^k is the MV concealment error and $\bar{P}^k(m)$ is the weighted average PEP of all residual packets in the k-th frame; β^k is the percentage of encoded I-MBs in the k-th frame; both the propagation factor α^k and the correlation ratio λ^k depend on video content, channel condition and codec structure, and are therefore called *system parameters*; D^{k-1} is the transmission distortion in the $k-1$ frame, which can be iteratively calculated by (1).

2.2.2. Transmission Distortion Formulae for PTD

From Ref. [1], we know that PTD can be calculated by

$$D^k_{\mathbf{u}} = D^k_{\mathbf{u}}(r) + D^k_{\mathbf{u}}(m) + D^k_{\mathbf{u}}(P) + D^k_{\mathbf{u}}(c), \tag{6}$$

where

$$D^k_{\mathbf{u}}(r) = E[(\varepsilon^k_{\mathbf{u}})^2] \cdot P^k_{\mathbf{u}}(r); \tag{7}$$

$$D^k_{\mathbf{u}}(m) = E[(\xi^k_{\mathbf{u}})^2] \cdot P^k_{\mathbf{u}}(m); \tag{8}$$

$$D^k_{\mathbf{u}}(P) = P^k_{\mathbf{u}} \cdot D^{k-1}_{\mathbf{u}+\check{\mathbf{m}}\mathbf{v}^k_{\mathbf{u}}} + (1-P^k_{\mathbf{u}}) \cdot D^k_{\mathbf{u}}(p); \tag{9}$$

$$D^k_{\mathbf{u}}(c) = 2P^k_{\mathbf{u}} \cdot (2E[\varepsilon^k_{\mathbf{u}} \cdot \xi^k_{\mathbf{u}}] + 2E[\varepsilon^k_{\mathbf{u}} \cdot \widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{m}}\mathbf{v}^k_{\mathbf{u}}}] + 2E[\xi^k_{\mathbf{u}} \cdot \widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{m}}\mathbf{v}^k_{\mathbf{u}}}]); \tag{10}$$

where $D^k_{\mathbf{u}}(p) \triangleq E[(\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\})^2]$ for $j \in \{1,...,J\}$; J is the number of previous encoded frames used for inter motion search; $\widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\}$ denotes the clipping noise under the error event that the packet is correctly received.

3. Algorithms for Estimating FTD

In this section, we develop our algorithms for estimating FTD under two scenarios: one without acknowledgement feedback and one with acknowledgement feedback, which are presented in Sections 3.1 and 3.2, respectively.

3.1. FTD Estimation without Feedback Acknowledgement

Next, Sections 3.1.1 through 3.1.4 present methods to estimate each of the four distortion terms in (1), respectively.

3.1.1. Estimation of Residual Caused Distortion

From the analysis in Ref. [1], $E[(\varepsilon^k)^2] = E[(\varepsilon^k_{\mathbf{u}})^2] = E[(\hat{e}^k_{\mathbf{u}} - \check{e}^k_{\mathbf{u}})^2]$ for all $\mathbf{u}$ in the k-th frame; $\hat{e}^k_{\mathbf{u}}$ is the transmitted residual for pixel $\mathbf{u}^k$; and $\check{e}^k_{\mathbf{u}}$ is the concealed residual for pixel $\mathbf{u}^k$ at the decoder. $E[(\varepsilon^k)^2]$ can be estimated from the finite samples of $\varepsilon^k_{\mathbf{u}}$ in the k-th frame, i.e., $\hat{E}[(\varepsilon^k)^2] = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}(\hat{e}^k_{\mathbf{u}} - \widehat{\check{e}^k_{\mathbf{u}}})^2$, where $\widehat{\check{e}^k_{\mathbf{u}}}$ is the estimate of $\check{e}^k_{\mathbf{u}}$.

From the analysis in Ref. [1], $\bar{P}^k(r) = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{N^k(r)}(P^k_i(r)\cdot N^k_i(r))$, where $P^k_i(r)$ is the PEP of the i-th residual packet in the k-th frame; $N^k_i(r)$ is the number of pixels contained in the i-th residual packet of the k-th frame; $N^k(r)$ is the number of residual packets in the k-th frame. $P^k_i(r)$ can be estimated from channel state statistics. Denote the estimated PEP by $\hat{P}^k_i(r)$ for all $i \in \{1, 2, ..., N^k(r)\}$; then $\bar{P}^k(r)$ can be estimated by $\hat{P}^k(r) = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{N^k(r)}(\hat{P}^k_i(r)\cdot N^k_i(r))$. As a result, $D^k(r)$ can be estimated by

$$\hat{D}^k(r) = \hat{E}[(\varepsilon^k)^2]\cdot\hat{P}^k(r) = \frac{1}{(|\mathcal{V}|)^2}\sum_{i=1}^{N^k(r)}(\hat{P}^k_i(r)\cdot N^k_i(r))\sum_{\mathbf{u}\in\mathcal{V}^k}(\hat{e}^k_{\mathbf{u}} - \widehat{\check{e}^k_{\mathbf{u}}})^2. \tag{11}$$

Appendix B presents how to 1) conceal $\hat{e}^k_{\mathbf{u}}$ at the decoder; 2) estimate $\check{e}^k_{\mathbf{u}}$ at the encoder; and 3) estimate $P^k_i(r)$ at the encoder.

3.1.2. Estimation of MV Caused Distortion

From the analysis in Ref. [1], $E[(\xi^k)^2] = E[(\xi^k_{\mathbf{u}})^2] = E[(\hat{f}^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} - \hat{f}^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}^k_{\mathbf{u}}})^2]$ for all $\mathbf{u}$ in the k-th frame; $\mathbf{mv}^k_{\mathbf{u}}$ is the transmitted MV for pixel $\mathbf{u}^k$; and $\check{\mathbf{mv}}^k_{\mathbf{u}}$ is the concealed MV for pixel $\mathbf{u}^k$ at the decoder. $E[(\xi^k)^2]$ can be estimated from the finite samples of $\xi^k_{\mathbf{u}}$ in the k-th frame, i.e., $\hat{E}[(\xi^k)^2] = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}(\hat{f}^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} - \hat{f}^{k-1}_{\mathbf{u}+\widehat{\check{\mathbf{mv}}^k_{\mathbf{u}}}})^2$, where $\widehat{\check{\mathbf{mv}}^k_{\mathbf{u}}}$ is the estimate of $\check{\mathbf{mv}}^k_{\mathbf{u}}$.

Similar to Section 3.1.1, $\bar{P}^k(m) = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{N^k(m)}(P^k_i(m)\cdot N^k_i(m))$, where $P^k_i(m)$ is the PEP of the i-th MV packet in the k-th frame; $N^k_i(m)$ is the number of pixels contained in the i-th MV packet of the k-th frame; $N^k(m)$ is the number of MV packets in the k-th frame. $P^k_i(m)$ can be estimated from channel state statistics. Denote the estimated PEP by $\hat{P}^k_i(m)$ for all $i \in \{1, 2, ..., N^k(m)\}$; then $\bar{P}^k(r)$ can be estimated by $\hat{P}^k(m) = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{N^k(m)}(\hat{P}^k_i(m)\cdot N^k_i(m))$. As a result, $D^k(m)$ can be estimated by

$$\hat{D}^k(m) = \hat{E}[(\xi^k)^2]\cdot\hat{P}^k(m) = \frac{1}{(|\mathcal{V}|)^2}\sum_{i=1}^{N^k(m)}(\hat{P}^k_i(m)\cdot N^k_i(m))\sum_{\mathbf{u}\in\mathcal{V}^k}(\hat{f}^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} - \hat{f}^{k-1}_{\mathbf{u}+\widehat{\check{\mathbf{mv}}^k_{\mathbf{u}}}})^2. \tag{12}$$

Appendix C presents how to 1) conceal $\mathbf{mv}^k_{\mathbf{u}}$ at the decoder; 2) estimate $\check{\mathbf{mv}}^k_{\mathbf{u}}$ at the encoder; and 3) estimate $P^k_i(m)$ at the encoder.

3.1.3. Estimation of Propagation and Clipping Caused Distortion

To estimate $D^k(P)$, we only need to estimate α^k since $\bar{P}^k(r)$ has been estimated in Section 3.1.1. In Ref. [1], we theoretically derive the propagation factor $\alpha^k_{\mathbf{u}}$ of pixel $\mathbf{u}^k$ for

propagated error with a zero-mean Laplacian distribution, i.e.,

$$\alpha = 1 - \frac{1}{2}e^{-\frac{y-\gamma_L}{b}}(\frac{y-\gamma_L}{b}+1) - \frac{1}{2}e^{-\frac{\gamma_H-y}{b}}(\frac{(\gamma_H-y)}{b}+1), \tag{13}$$

where γ_L and γ_H are user-specified low threshold and high threshold, respectively; y is the reconstructed pixel value; $b = \frac{\sqrt{2}}{2}\sigma$; and σ is the standard deviation of the propagated error.

Here, we provide three methods to estimate the propagation factor α^k as below.

Estimation of α^k by $\alpha^k_{\mathbf{u}}$: As defined in Ref. [1], $\alpha^k = \frac{\sum_{\mathbf{u}\in\mathcal{V}^k}\alpha^k_{\mathbf{u}}\cdot D^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}}{\sum_{\mathbf{u}\in\mathcal{V}^k} D^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}}$. Therefore, we may first estimate $\alpha^k_{\mathbf{u}}$ by (13) and then estimate α^k by its definition. However, this method requires to compute exponentiations and divisions in (13) for each pixel, and needs large memory to store $\hat{D}^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}$ for all pixels in all reference frames.

Estimate the average of a function by the function of an average: If we estimate α^k directly by the frame statistics instead of pixel values, both the computational complexity and memory requirement will be decreased by a factor of N_{V^k}. If only $\hat{D}^{k-1}$ instead of $\hat{D}^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}$ is stored in memory, we may simplify estimating α^k by $\hat{\alpha}^k = \frac{\sum_{\mathbf{u}\in\mathcal{V}^k}\hat{\alpha}^k_{\mathbf{u}}\cdot\hat{D}^{k-1}}{\sum_{\mathbf{u}\in\mathcal{V}^k}\hat{D}^{k-1}} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\hat{\alpha}^k_{\mathbf{u}}$. This is accurate if all packets in the same frame experience the same channel condition. We see from (13) that $\alpha^k_{\mathbf{u}}$ is a function of the reconstructed pixel value $\hat{f}^k_{\mathbf{u}}$ and the variance of propagated error $\sigma^2_{\hat{f}^{k-1}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}}$, which is equal to D^{k-1} in this case. Denote $\alpha^k_{\mathbf{u}} = g(\hat{f}^k_{\mathbf{u}}, D^{k-1})$; we have $\alpha^k = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k} g(\hat{f}^k_{\mathbf{u}}, D^{k-1})$. One simple and intuitive method is to use the function of an average to estimate the average of a function, that is, $\hat{\alpha}^k = g(\frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\hat{f}^k_{\mathbf{u}}, \hat{D}^{k-1})$.

Improve estimation accuracy by using the property of (13)**:** Although the above method dramatically reduces the estimation complexity and memory requirement, that simple approximation is only accurate if $\alpha^k_{\mathbf{u}}$ is a linear function of $\hat{f}^k_{\mathbf{u}}$. In other words, such approximation causes underestimation for the convex function or overestimation for the concave function [11]. Although (13) is neither a convex function nor a concave function, it is interesting to see that 1) $\alpha^k_{\mathbf{u}}$ is symmetric about $\hat{f}^k_{\mathbf{u}} = \frac{\gamma_H+\gamma_L}{2}$; 2) $\alpha^k_{\mathbf{u}}$ is a monotonically increasing function of $\hat{f}^k_{\mathbf{u}}$ when $\gamma_L < \hat{f}^k_{\mathbf{u}} < \frac{\gamma_H+\gamma_L}{2}$, and $\alpha^k_{\mathbf{u}}$ is a monotonically decreasing function of $\hat{f}^k_{\mathbf{u}}$ when $\frac{\gamma_H+\gamma_L}{2} < \hat{f}^k_{\mathbf{u}} < \gamma_H$; 3) both half sides are much more linear than the whole function. So, we propose to use $\frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}|\hat{f}^k_{\mathbf{u}} - \frac{\gamma_H+\gamma_L}{2}| + \frac{\gamma_H+\gamma_L}{2}$ instead of $\frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\hat{f}^k_{\mathbf{u}}$ to estimate α^k. Since the symmetry property is exploited, such algorithm gives much more accurate estimate $\hat{\alpha}^k$.

From the analysis in Ref. [1], we have $D^k(p) = \alpha^k \cdot D^{k-1}$; so we can estimate $D^k(p)$ by $\hat{D}^k(p) = \hat{D}^{k-1}\cdot\hat{\alpha}^k$. To compensate the accuracy loss of using frame statistics, we may use the following algorithm to estimate $D^k(p)$ without the exponentiation and division for each pixel:

$$\hat{D}^k(p) = (\hat{D}^{k-1} - \hat{D}^{k-1}(r) - \hat{D}^{k-1}(m))\cdot\hat{\alpha}^k + \overline{\Phi^2(\varepsilon^{k-1}, \hat{f}^k)} + \overline{\Phi^2(\xi^{k-1}, \hat{f}^k)}, \tag{14}$$

where $\hat{D}^{k-1}(r)$ can be estimated by (11); $\hat{D}^{k-1}(m)$ can be estimated by (12); $\hat{\alpha}^k$ can be estimated by (13); $\overline{\Phi^2(\varepsilon^{k-1},\, \hat{f}^k)} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\Phi^2(\varepsilon^{k-1}_{\mathbf{u}},\, \hat{f}^k_{\mathbf{u}})$ and $\overline{\Phi^2(\xi^{k-1},\, \hat{f}^k)} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\Phi^2(\xi^{k-1}_{\mathbf{u}},\, \hat{f}^k_{\mathbf{u}})$, while both of them can be easily calculated by

$$\Phi(x,\, y) \triangleq y - \Gamma(y-x) = \begin{cases} y-\gamma_L, & y-x < \gamma_L \\ x, & \gamma_L \le y-x \le \gamma_H \\ y-\gamma_H, & y-x > \gamma_H. \end{cases} \tag{15}$$

Our experimental results in Section 7 show that the proposed algorithm provides accurate estimate. Finally, it is straightforward to estimate $D^k(P)$ by

$$\hat{D}^k(P) = \hat{P}^k(r)\cdot\hat{D}^{k-1} + (1-\beta^k)\cdot(1-\hat{P}^k(r))\cdot\hat{D}^k(p). \tag{16}$$

3.1.4. Estimation of Correlation-Caused Distortion

To estimate $D^k(c)$, the only parameter needs to be estimated is λ^k since $D^k(m)$ has been estimated in Section 3.1.2. As defined in Ref. [1], $\lambda^k = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\lambda^k_{\mathbf{u}}$, where $\lambda^k_{\mathbf{u}} = \frac{E[\xi^k_{\mathbf{u}}\cdot\tilde{f}^{k-1}_{\mathbf{u}+\tilde{\mathbf{mv}}^k_{\mathbf{u}}}]}{E[\xi^k_{\mathbf{u}}\cdot\hat{f}^{k-1}_{\mathbf{u}+\tilde{\mathbf{mv}}^k_{\mathbf{u}}}]}$. $\lambda^k_{\mathbf{u}}$ depends on the motion activity of the video content according to Ref. [1].

In our experiment, we find that λ^k is small when the average MV length over the set in the k-th frame is larger than half of the block length, and $\lambda^k \approx 1$ when the average MV length in the k-th frame is smaller than half of the block length, or when the propagated error from the reference frames is small. An intuitive explanation for this phenomenon is as below: 1) if the average MV length is large and the MV packets are received with error, most concealed reference pixels will be in some block different from the block where the corresponding true reference pixels locate; 2) if the average MV length is small, most concealed reference pixels and the corresponding true reference pixels will still be in the same block even if the MV packet is received with error; 3) since the correlation between two pixels inside the same block is much higher than the correlation between two pixels located in different blocks, hence λ^k is small when the average MV length is large and vice versa; 4) if there is no propagated error from the reference frames, according to the definition, it is easy to prove that $\lambda^k = 1$.

Therefore, we propose a low complexity algorithm to estimate λ^k by frame statistics as below

$$\hat{\lambda}^k = \begin{cases} (1-\bar{P}^{k-1}(m))(1-\bar{P}^{k-1}(r)), & \overline{|\mathbf{mv}^k|} > \frac{\text{block\_size}}{2} \\ 1, & \text{otherwise}, \end{cases} \tag{17}$$

where $\bar{P}^{k-1}(r)$ is defined in (4); $\bar{P}^{k-1}(m)$ is defined in (5); $\overline{|\mathbf{mv}^k|} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}|\mathbf{mv}^k_{\mathbf{u}}|$, and $|\mathbf{mv}^k_{\mathbf{u}}|$ is the length of $\mathbf{mv}^k_{\mathbf{u}}$. As a result,

$$\hat{D}^k(c) = (\hat{\lambda}^k - 1)\cdot\hat{D}^k(m). \tag{18}$$

3.1.5. Summary

Without feedback acknowledgement, the transmission distortion of the k-th frame can be estimated by

$$\hat{D}^k = \hat{D}^k(r) + \hat{\lambda}^k \cdot \hat{D}^k(m) + \hat{P}^k(r) \cdot \hat{D}^{k-1} + (1-\beta^k) \cdot (1-\hat{P}^k(r)) \cdot \hat{D}^k(p), \qquad (19)$$

where $\hat{D}^k(r)$ can be estimated by (11); $\hat{D}^k(m)$ can be estimated by (12); $\hat{D}^k(p)$ can be estimated by (14); $\hat{\lambda}^k$ can be estimated by (17); $\hat{P}^k(r)$ can be estimated by the estimated PEP of all residual packets in the k-th frame as discussed in Section 3.1.1. We call the resulting algorithm in (19) as **RMPC-FTD algorithm**.

3.2. FTD Estimation with Feedback Acknowledgement

In some wireless video communication systems, the receiver may send the transmitter a notification about whether packets are correctly received. This feedback acknowledgement mechanism can be utilized to improve FTD estimation accuracy as shown in Algorithm 1.

Algorithm 1. *FTD estimation at the transmitter under feedback acknowledgement.*

1) ***Input****:* $\hat{P}_i^k(r)$ *and* $\hat{P}_i^k(m)$ *for all* $i \in \{1, 2, ..., N^k\}$.
2) Initialization and update.
If $k = 1$*, do initialization.*
If $k > 1$*, update with feedback information.*
If there are acknowledgements for packets in the $(k-1)$*-th frame,*
For $j = 1 : N^{k-1}$
if ACK for the j*-th residual packet is received, update* $\hat{P}_j^{k-1}(r) = 0$.
if NACK for the j*-th residual packet is received, update* $\hat{P}_j^{k-1}(r) = 1$.
if ACK for the j*-th MV packet is received, update* $\hat{P}_j^{k-1}(m) = 0$.
if NACK for the j*-th MV packet is received, update* $\hat{P}_j^{k-1}(m) = 1$.
End
Update $\hat{D}^{k-1}$.
Else (neither ACK nor NACK is received), go to 3).
3) Estimate D^k *via*
$\hat{D}^k = \hat{D}^k(r) + \hat{\lambda}^k \cdot \hat{D}^k(m) + \hat{P}^k(r) \cdot \hat{D}^{k-1} + (1-\beta^k) \cdot (1-\hat{P}^k(r)) \cdot \hat{D}^k(p)$,
which is (19).
4) ***Output****:* $\hat{D}^k$.

Algorithm 1 has a low computational complexity since $\hat{D}^{k-1}$ is updated based on whether packets in the $(k-1)$-th frame are correctly received or not. In a more general case that the encoder can tolerate a feedback delay of d frames, we could update $\hat{D}^{k-1}$ based on the feedback acknowledgements for the $(k-d)$-th frame through the $(k-1)$-th frame. However, this requires extra memory for the encoder to store all the system parameters from the $(k-d)$-th frame to the $(k-1)$-th frame in order to update $\hat{D}^{k-1}$.

4. Pixel-level Transmission Distortion Estimation Algorithm

The PTD estimation algorithm is similar to the FTD estimation algorithm presented in Section 3. However, the values of some variables in the PTD formula derived in Ref. [1] may be known at the encoder. Taking $\mathbf{u}^k$ as an example, before the prediction mode is selected, the best motion vector $\mathbf{mv}_{\mathbf{u}}^k$ of each prediction mode is known after motion estimation is done; hence the residual $\hat{e}_{\mathbf{u}}^k$ and reconstructed pixel value $\hat{f}_{\mathbf{u}}^k$ of each mode are also known. In such a case, these known values could be used to replace the statistics of the corresponding random variables to simplify the PTD estimation. In this section, we discuss how to use the known values to improve the estimation accuracy and reduce the algorithm complexity.

4.1. Estimation of PTD

In this section, we consider the case with no data partitioning; hence, $P_{\mathbf{u}}^k = P_{\mathbf{u}}^k(r) = P_{\mathbf{u}}^k(m)$. For the case with slice data partitioning, the derivation process is similar to that in this section.

In (6)-(10), if the values for $\mathbf{mv}_{\mathbf{u}}^k$, $\hat{e}_{\mathbf{u}}^k$ and $\hat{f}_{\mathbf{u}}^k$ are known, given the error concealment at the encoder, the values for $\varepsilon_{\mathbf{u}}^k = \hat{e}_{\mathbf{u}}^k - \check{e}_{\mathbf{u}}^k$ and $\xi_{\mathbf{u}}^k = \hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} - \hat{f}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}$ are also known. Then, $D_{\mathbf{u}}^k(r) = (\varepsilon_{\mathbf{u}}^k)^2 \cdot P_{\mathbf{u}}^k$, $D_{\mathbf{u}}^k(m) = (\xi_{\mathbf{u}}^k)^2 \cdot P_{\mathbf{u}}^k$, and $D_{\mathbf{u}}^k(c) = P_{\mathbf{u}}^k \cdot (2\varepsilon_{\mathbf{u}}^k \cdot \xi_{\mathbf{u}}^k + 2\varepsilon_{\mathbf{u}}^k \cdot E[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}] + 2\xi_{\mathbf{u}}^k \cdot E[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}])$. Hence, the formula for PTD can be simplified to

$$D_{\mathbf{u}}^k = E[(\widetilde{\zeta}_{\mathbf{u}}^k)^2] = P_{\mathbf{u}}^k \cdot ((\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k)^2 + 2(\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k) \cdot E[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}] + D_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}) + (1 - P_{\mathbf{u}}^k) \cdot D_{\mathbf{u}}^k(p). \tag{20}$$

Denote $\hat{D}(\cdot)$ the estimate of $D(\cdot)$, and denote $\hat{E}(\cdot)$ as the estimate of $E(\cdot)$. Therefore, $D_{\mathbf{u}}^k$ can be estimated by $\hat{D}_{\mathbf{u}}^k = \hat{P}_{\mathbf{u}}^k \cdot ((\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k)^2 + 2(\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k) \cdot \hat{E}[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}] + \hat{D}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}) + (1 - \hat{P}_{\mathbf{u}}^k) \cdot \hat{D}_{\mathbf{u}}^k(p)$, where $\hat{P}_{\mathbf{u}}^k$ can be obtained by the PEP estimation algorithm in Section 3. $\hat{D}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}$ is the estimate in the $(k-1)$-th frame and is stored for calculating $D_{\mathbf{u}}^k$. Therefore, the only unknowns are $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}]$ and $\hat{D}_{\mathbf{u}}^k(p)$, which can be calculated by the methods in Sections 4.2 and 4.3.

4.2. Calculation of $\hat{E}[\widetilde{\zeta}_{\mathbf{u}}^k]$

Since $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}]$ from the $(k-1)$-th frame is required for calculating $\hat{D}_{\mathbf{u}}^k$, we should estimate the first moment of $\widetilde{\zeta}_{\mathbf{u}}^k$ and store it for the subsequent frame. From Ref. [1], we know $\widetilde{\zeta}_{\mathbf{u}}^k = \widetilde{\varepsilon}_{\mathbf{u}}^k + \widetilde{\xi}_{\mathbf{u}}^k + \widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-j'} + \widetilde{\Delta}_{\mathbf{u}}^k$. For P-MBs, when MV packet is correctly received, $\widetilde{\varepsilon}_{\mathbf{u}}^k = \widetilde{\xi}_{\mathbf{u}}^k = 0$ and $\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-j'} = \widetilde{\zeta}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-j}$; when MV packet is received with error, $\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-j'} = \widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}$, and since residual and MV are in the same packet, $\widetilde{\Delta}_{\mathbf{u}}^k\{r, m\} = \widetilde{\Delta}_{\mathbf{u}}^k\{r\} = 0$ as proved in Ref. [1]. Therefore, the first moment of $\zeta_{\mathbf{u}}^k$ can be recursively calculated by

$$E[\widetilde{\zeta}_{\mathbf{u}}^k] = P_{\mathbf{u}}^k \cdot (\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k + E[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}]) + (1 - P_{\mathbf{u}}^k) \cdot E[\widetilde{\zeta}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}]. \tag{21}$$

Consequently, $E[\widetilde{\zeta}_{\mathbf{u}}^k]$ can be estimated by $\hat{E}[\widetilde{\zeta}_{\mathbf{u}}^k] = \hat{P}_{\mathbf{u}}^k \cdot (\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k + \hat{E}[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{m}}\mathbf{v}_{\mathbf{u}}^k}^{k-1}]) + (1 - \hat{P}_{\mathbf{u}}^k) \cdot \hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}]$.

For I-MBs, when the packet is correctly received, $\widetilde{\zeta}_{\mathbf{u}}^k = 0$; when MV packet is received with error, the result is the same as for P-MBs. Therefore, the first moment of $\zeta_{\mathbf{u}}^k$ can be recursively calculated by

$$E[\widetilde{\zeta}_{\mathbf{u}}^k] = P_{\mathbf{u}}^k \cdot (\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k + E[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{m}}\mathbf{v}_{\mathbf{u}}^k}^{k-1}]), \tag{22}$$

and $E[\widetilde{\zeta}_{\mathbf{u}}^k]$ can be estimated by $\hat{E}[\widetilde{\zeta}_{\mathbf{u}}^k] = \hat{P}_{\mathbf{u}}^k \cdot (\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k + \hat{E}[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{m}}\mathbf{v}_{\mathbf{u}}^k}^{k-1}])$.

4.3. Calculation of $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}]$ and $\hat{D}_{\mathbf{u}}^k(p)$

From Ref. [1], we know that for I-MBs, $D_{\mathbf{u}}^k(p) = 0$; for P-MBs, $D_{\mathbf{u}}^k(p) = \alpha_{\mathbf{u}}^k \cdot D_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}$ and it can be estimated by $\hat{D}_{\mathbf{u}}^k(p) = \hat{\alpha}_{\mathbf{u}}^k \cdot \hat{D}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}$, where $\hat{\alpha}_{\mathbf{u}}^k$ is estimated by (13) with $y = \hat{f}_{\mathbf{u}}^k$ and $\sigma^2 = \hat{D}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}$.

However, such complexity is too high to be used in prediction mode decision since every pixel requires such a computation for each mode. To address this, we leverage the property proved in Proposition 1 to design a low-complexity and high-accuracy algorithm to recursively calculate $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}]$ and $\hat{D}_{\mathbf{u}}^k(p)$ for P-MBs.

Proposition 1. *Assume $\gamma_H = 255$ and $\gamma_L = 0$. The propagated error $\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}$ and the clipping noise $\widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}$ satisfy*

$$\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\} = \begin{cases} \hat{f}_{\mathbf{u}}^k - 255, & \widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} < \hat{f}_{\mathbf{u}}^k - 255 \\ \hat{f}_{\mathbf{u}}^k, & \widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} > \hat{f}_{\mathbf{u}}^k \\ \widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}, & \textit{otherwise.} \end{cases} \tag{23}$$

Proposition 1 is proved in Appendix D. Using Proposition 1, $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}]$ in (21) and $\hat{D}_{\mathbf{u}}^k(p)$ in (20) can be estimated under the following three cases.

Case 1: If $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}] < \hat{f}_{\mathbf{u}}^k - 255$, we have $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}] = \hat{f}_{\mathbf{u}}^k - 255$, and $\hat{D}_{\mathbf{u}}^k(p) = \hat{E}[(\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\})^2] = (\hat{f}_{\mathbf{u}}^k - 255)^2$.

Case 2: If $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}] > \hat{f}_{\mathbf{u}}^k$, we have $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}] = \hat{f}_{\mathbf{u}}^k$, and $\hat{D}_{\mathbf{u}}^k(p) = (\hat{f}_{\mathbf{u}}^k)^2$.

Case 3: If $\hat{f}_{\mathbf{u}}^k - 255 \le \hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}] \le \hat{f}_{\mathbf{u}}^k$, we have $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}] = \hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j}]$, and $\hat{D}_{\mathbf{u}}^k(p) = \hat{E}[(\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j})^2]$.

4.4. Summary

PTD can be recursively estimated by (20) and (21) or (22); and $\hat{E}[\widetilde{\zeta}_{\mathbf{u}+\mathbf{m}\mathbf{v}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r}, \bar{m}\}]$ and $\hat{D}_{\mathbf{u}}^k(p)$ can be calculated by the methods in Section 4.3. The resulting algorithm is called **RMPC-PTD** algorithm.

5. Pixel-level End-to-end Distortion Estimation Algorithm

The pixel-level end-to-end distortion (PEED) for each pixel $\mathbf{u}$ in the k-th frame is defined by $D^k_{\mathbf{u},ETE} \triangleq E[(f^k_{\mathbf{u}} - \widetilde{f}^k_{\mathbf{u}})^2]$, where $f^k_{\mathbf{u}}$ is the input pixel value at the encoder and $\widetilde{f}^k_{\mathbf{u}}$ is the reconstructed pixel value at the decoder. Then we have

$$\begin{aligned} D^k_{\mathbf{u},ETE} &= E[(f^k_{\mathbf{u}} - \widetilde{f}^k_{\mathbf{u}})^2] \\ &= E[(f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}} + \hat{f}^k_{\mathbf{u}} - \widetilde{f}^k_{\mathbf{u}})^2] \\ &= E[(f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}} + \widetilde{\zeta}^k_{\mathbf{u}})^2] \\ &= (f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}})^2 + E[(\widetilde{\zeta}^k_{\mathbf{u}})^2] + 2(f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}}) \cdot E[\widetilde{\zeta}^k_{\mathbf{u}}]. \end{aligned} \tag{24}$$

We call $f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}}$ quantization error and $\widetilde{\zeta}^k_{\mathbf{u}}$ transmission error. While $f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}}$ depends only on the quantization parameter (QP), $\widetilde{\zeta}^k_{\mathbf{u}}$ mainly depends on the PEP and the error concealment scheme. If the value of $\hat{f}^k_{\mathbf{u}}$ is known, then the only unknowns in (24) are $E[(\widetilde{\zeta}^k_{\mathbf{u}})^2]$ and $E[\widetilde{\zeta}^k_{\mathbf{u}}]$, which can be estimated by the methods in Section 4. We call the algorithm in (24) as **RPMC-PEED** algorithm.

Compared to ROPE algorithm [8], which estimates the first moment and second moment of the reconstructed pixel value $\widetilde{f}^k_{\mathbf{u}}$, we have the following observations. First, RPMC-PEED algorithm estimates the first moment and the second moment of reconstructed error $\widetilde{\zeta}^k_{\mathbf{u}}$; therefore, RPMC-PEED algorithm is much easier to be enhanced to support the averaging operations in H.264, such as interpolation filter. Second, estimating the first moment and the second moment of $\widetilde{\zeta}^k_{\mathbf{u}}$ in RMPC-PEED produces lower distortion estimation error than estimating both moments of $\widetilde{f}^k_{\mathbf{u}}$ in ROPE. Third, our experimental results show that ROPE may produce a negative value as the estimate for distortion, which violates the requirement that (true) distortion must be non-negative; our experimental results also show that the negative distortion estimate is caused by not considering clipping, which results in inaccurate distortion estimation by ROPE.

Note that in Ref. [1], we assume the clipping noise at the encoder is zero, that is, $\hat{\Delta}^k_{\mathbf{u}} = 0$. If we use $\hat{f}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \hat{e}^k_{\mathbf{u}}$ to replace $\hat{f}^k_{\mathbf{u}}$ in (24), we may calculate the quantization error by $f^k_{\mathbf{u}} - (\hat{f}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \hat{e}^k_{\mathbf{u}})$ and calculate the transmission error by

$$\begin{aligned} \widetilde{\zeta}^k_{\mathbf{u}} &= (\hat{f}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \hat{e}^k_{\mathbf{u}}) - \widetilde{f}^k_{\mathbf{u}} \\ &= (\hat{f}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \hat{e}^k_{\mathbf{u}}) - (\widetilde{f}^{k-1}_{\mathbf{u}+\widetilde{\mathbf{mv}}^k_{\mathbf{u}}} + \widetilde{e}^k_{\mathbf{u}} - \widetilde{\Delta}^k_{\mathbf{u}}) \\ &= \widetilde{\varepsilon}^k_{\mathbf{u}} + \widetilde{\xi}^k_{\mathbf{u}} + \widetilde{\zeta}^{k-1}_{\mathbf{u}+\widetilde{\mathbf{mv}}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}, \end{aligned} \tag{25}$$

which is exactly the formula for transmission error decomposition in Ref. [1]. Therefore, $\hat{\Delta}^k_{\mathbf{u}}$ does not affect the end-to-end distortion $D^k_{\mathbf{u},ETE}$ if we use $\hat{f}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \hat{e}^k_{\mathbf{u}}$ to replace $\hat{f}^k_{\mathbf{u}}$ in calculating both the quantization error and the transmission error.

6. Applying RMPC-PEED Algorithm to H.264 Prediction Mode Decision

6.1. Rate-distortion Optimized Prediction Mode Decision

In H.264 specification, there are two types of prediction modes, i.e., inter prediction and intra prediction[2]. In inter prediction, there are 7 modes, i.e., modes for 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4 luma blocks. In intra prediction, there are 9 modes for 4x4 luma blocks and 4 modes for 16x16 luma blocks. Hence, there are a total of $7+9+4=20$ modes to be selected in mode decision. For each MB, our proposed Error-Resilient Rate Distortion Optimized (ERRDO) mode decision consists of two steps. First, R-D cost is calculated by

$$J(\omega_m) = D^k_{ETE}(\omega_m) + \lambda \cdot R(\omega_m), \tag{26}$$

where $D^k_{ETE} = \sum_{\mathbf{u}\in\mathcal{V}^k_i} D^k_{\mathbf{u},ETE}$; $\mathcal{V}^k_i$ is the set of pixels in the i-th MB (or sub-MB) of the k-th frame; ω_m is the prediction mode, and ω_m ($\omega_m \in \{1, 2, \cdots, 20\}$); $R(\omega_m)$ is the encoded bit rate for mode ω_m; λ is the preset Lagrange multiplier. Then, the optimal prediction mode that minimizes the rate-distortion (R-D) cost is found by

$$\hat{\omega}_m = \arg\min_{\omega_m}\{J(\omega_m)\}. \tag{27}$$

If $D^k_{ETE}(\omega_m)$ in (26) is replaced by source coding distortion or quantization distortion, we call it Source-Coding Rate Distortion Optimized (SCRDO) mode decision.

Using (26) and (27), we design Algorithm 2 for ERRDO mode decision in H.264; Algorithm 2 is called **RMPC-MS** algorithm.

Algorithm 2. *ERRDO Mode decision for an MB in the k-th frame ($k \geq 1$).*

1) ***Input****: QP, PEP.*
2) Initialization of $\hat{E}[\widetilde{\zeta}^0_{\mathbf{u}}]$ *and* $\hat{E}[(\widetilde{\zeta}^0_{\mathbf{u}})^2]$ *for all pixel* $\mathbf{u}$.
3) For mode = 1 : 20 (9+4 intra, 7 inter).
  3a) If intra mode,
    calculate $\hat{E}[\widetilde{\zeta}^k_{\mathbf{u}}]$ *by (22) for all pixels in the MB,*
    go to 3b),
  Else if $\hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}] < \hat{f}^k_{\mathbf{u}} - 255$,
    $\hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\}] = \hat{f}^k_{\mathbf{u}} - 255$,
    $\hat{E}[(\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\})^2] = (\hat{f}^k_{\mathbf{u}} - 255)^2$,
  Else if $\hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}] > \hat{f}^k_{\mathbf{u}}$
    $\hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\}] = \hat{f}^k_{\mathbf{u}}$,
    $\hat{E}[(\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\})^2] = (\hat{f}^k_{\mathbf{u}})^2$,
  Else

[2]There are two other encoding modes for P-MB defined in H.264, i.e., skip mode and I_PCM mode. However, they are usually not involved in the PEED estimation process.

$\hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\}] = \hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}]$,
$\hat{E}[(\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\})^2] = \hat{E}[(\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}})^2]$,
End
calculate $\hat{E}[\widetilde{\zeta}^k_{\mathbf{u}}]$ *by (21) for all pixels in the MB,*
3b) calculate $\hat{D}^k_{\mathbf{u}}$ *by (20) for all pixels in the MB,*
3c) estimate $D^k_{\mathbf{u},ETE}$ *by (24) for all pixels in the MB,*
3d) calculate R-D cost via (26) for each mode,
End
Via (27)*, select the mode with minimum R-D cost as the optimal mode for the MB.*
5) ***Output****: the best mode for the MB.*

Using Theorem 1, we can design another ERRDO mode decision algorithm that produces the same solution as that of Algorithm 2, as Proposition 2 states.

Theorem 1. *(Decomposition Theorem) If there is no slice data partitioning, end-to-end distortion can be decomposed into a mode-dependent term and a mode-independent term, i.e.,*

$$D^k_{\mathbf{u},ETE}(\omega_m) = D^k_{\mathbf{u},\overline{ETE}}(\omega_m) + C^k_{\mathbf{u}}. \tag{28}$$

where $C^k_{\mathbf{u}}$ *is independent of* ω_m *and*

$$D^k_{\mathbf{u},\overline{ETE}}(\omega_m) = (1-P^k_{\mathbf{u}})\cdot\{(f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}})^2 + D^k_{\mathbf{u}}(p) + 2(f^k_{\mathbf{u}} - \hat{f}^k_{\mathbf{u}})\cdot E[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r},\bar{m}\}]\}. \tag{29}$$

Theorem 1 is proved in Appendix E.

Using Theorem 1, we only need to change two places in Algorithm 2 to obtain a new algorithm, which we call Algorithm A: first, replace Step 3c) in Algorithm 2 by "estimate $D^k_{\mathbf{u},\overline{ETE}}$ by (29) for all pixels in the MB"; second, replace 3d) in Algorithm 2 by "calculate R-D cost via $D^k_{\overline{ETE}}(\omega_m) + \lambda\cdot R(\omega_m)$ for each mode", where $D^k_{\overline{ETE}} = \sum_{\mathbf{u}\in\mathcal{V}^k_i} D^k_{\mathbf{u},\overline{ETE}}$.

Proposition 2. *If there is no slice data partitioning, Algorithm A and Algorithm 2 produce the same solution, i.e.,* $\hat{\omega}_m = \arg\min_{\omega_m}\{\hat{D}^k_{ETE}(\omega_m) + \lambda\cdot R(\omega_m)\} = \arg\min_{\omega_m}\{\hat{D}^k_{\overline{ETE}}(\omega_m) + \lambda\cdot R(\omega_m)\}$.

Proposition 2 is proved in Appendix F.

Note that $D^k_{\overline{ETE}}$ in (29) is not exactly the end-to-end distortion; but the decomposition in (28) can help reduce the complexity of some estimation algorithms, for example, LLN algorithm [12].

Table 1: Complexity Comparison

Algorithms		computational complexity	memory requirement
RMPC-MS	inter mode	9 ADDs, 8 MULs	25 bits/pixel
	intra mode	7 ADDs, 6 MULs	
	total complexity	154 ADDs, 134 MULs	
ROPE	inter mode	7 ADDs, 8 MULs	24 bits/pixel
	intra mode	4 ADDs, 7 MULs	
	total complexity	101 ADDs, 147 MULs	
LLN	inter mode	$2N_d$ ADDs, N_d MULs	$8N_d$ bits/pixel
	intra mode	N_d ADDs, N_d MULs	
	total complexity	$27N_d$ ADDs, $20N_d$ MULs	

6.2. Complexity of RMPC-MS, ROPE, and LLN Algorithm

In this subsection, we compare the complexity of RMPC-MS algorithm with that of two popular mode decision algorithms, namely, ROPE algorithm and LLN algorithm, which are also based on pixel-level distortion estimation. To make a fair comparison, the same conditions should be used for all the three algorithms. Assume all the three algorithms use an error concealment scheme that conceals an erroneous pixel by the pixel in the same position of the previous frame; then, $\check{e}^k_{\mathbf{u}} = 0$ and $\check{\mathbf{mv}}^k_{\mathbf{u}} = 0$; hence, $\varepsilon^k_{\mathbf{u}} + \xi^k_{\mathbf{u}} = f^k_{\mathbf{u}} - \hat{f}^{k-1}_{\mathbf{u}}$.

Here, the complexity is quantified by the number of additions (ADDs) and multiplications (MULs)[3]. If a subroutine (or the same set of operations) is invoked multiple times, it is counted only once since the temporary result is saved in the memory; for example, $\varepsilon^k_{\mathbf{u}} + \xi^k_{\mathbf{u}}$ in (20) and (21) is counted as one ADD. A substraction is counted as an addition. We only consider pixel-level operations; block-level operations, for example MV addition, are neglected. We ignore the complexity of those basic operations since their complexity is the same for all the three algorithms, such as motion compensation.

Table 1 shows the complexity of the three algorithms. Appendix G presents the details on how to calculate the computational complexity and memory requirement.

7. Experimental Results

In Section 7.1, we compare the estimation accuracy of RMPC-FTD algorithm to that of the existing models under different channel conditions; we also compare their robustness against imperfect estimate of PEP. Since the number of pixels in a frame is too large to plot the the estimation accuracy for each of them, in Section 7.2, we only compare the R-D performance of RMPC-MS to that of mode decision by existing PEED algorithms for H.264; we also compare them under interpolation filter and deblocking filter.

To analyze the statistics and verify our algorithm, our lab has developed a software tool, called Video Distortion Analysis Tool (VDAT), which provides a friendly Graphical User Interface (GUI). VDAT implements 1) channel simulator, 2) supports different video codec,

[3]Those minor operations, such as memory copy, shift, and conditional statement, are neglected for all algorithms.

3) computes the video statistics, e.g. the joint pdf of MV concealment error and propagated error, and 4) supports several distortion estimation algorithms, 5) compare R-D performance of different algorithms[4]. VDAT is used in all the experiments in this section.

7.1. Estimation Accuracy and Robustness

In this section, we use Algorithm 1 to estimate FTD and compare it with Stuhlmuller's model [2] and Dani's model [3]. To evaluate estimation accuracy, we compare the estimated distortion of different algorithms with true distortion for 50 frames under the case of no acknowledgement feedback.

7.1.1. Experiment Setup

To implement the estimation algorithms, all transmission distortion related statistics should be collected for all random variables, such as residual, motion vector, reconstructed pixel value, residual concealment error, MV concealment error, propagated error, clipping noise. All such statistics are collected from video codec JM14.0 [7]. All tested video sequences [13] are in CIF format, and each frame is divided into three slices. To support the slice data partitioning, we use the extended profile as defined in H.264 specification Annex A [14]. To provide unequal error protection (UEP), we let MV packets experience lower PEP than residual packets. The first frame of each coded video sequence is an I-frame, and the subsequent frames are all P-frames. In the experiment, we let the first I-frame be correctly received, and all the following P-frames go through an error-prone channel with controllable PEP. We set QP=28 for all the frames.

Each video sequence is tested under several channel conditions with UEP. Due to the space limit, we only present the experimental results for video sequences 'stefan' and 'foreman' in this paper[5]. For each sequence, two wireless channel conditions are tested: for good channel condition, residual PEP is 2% and MV PEP is 1%; for poor channel condition, residual PEP is 10% and MV PEP is 5%. Note that the packet error state is a two-value random variable, that is, it is a Bernoulli distributed with success probability $p = PEP$. For each PEP setting of each frame, we do 600 simulations and take the average to mitigate the effect of randomness of simulated channels on instantaneous distortion.

7.1.2. Estimation Accuracy of Different Estimation Algorithms

Fig. 2 shows the estimation accuracy of RMPC-FTD algorithm, Stuhlmuller's model in Ref. [2] and Dani's model in Ref. [3] for sequence 'stefan'. Fig. 3 shows their estimation accuracy for sequence 'foreman'. We can see that RMPC-FTD algorithm achieves the most accurate estimate.

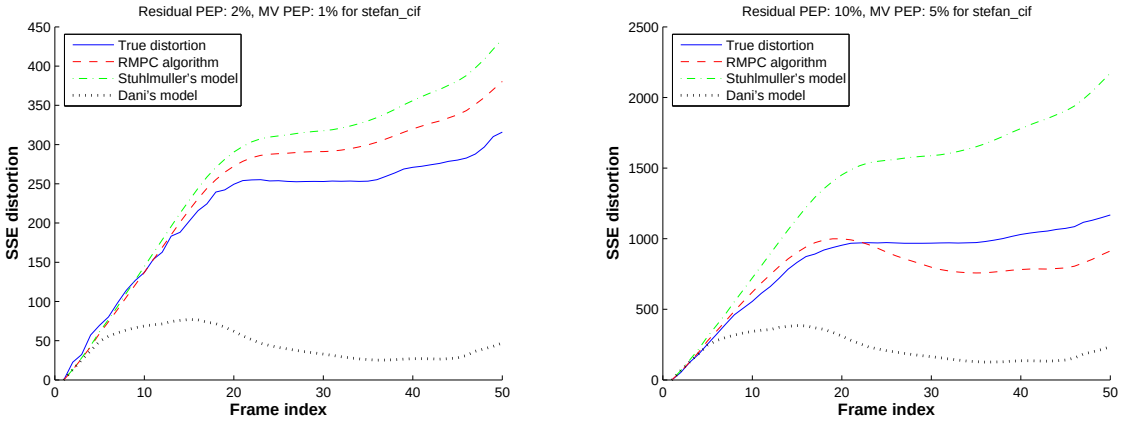
Since the superposition algorithm in Stuhlmuller's model neglects the effect of clipping noise and negative correlation between MV concealment error and propagated error, it overestimates transmission distortion as shown in Fig. 2. However, since the clipping effect and

[4]Please refer to http://www.mcn.ece.ufl.edu/public/zhifeng/project/VDAT for more detail information.

[5]Experimental results for other video sequences can be found at http://www.mcn.ece.ufl.edu/public/zhifeng/project/VDAT/journal.

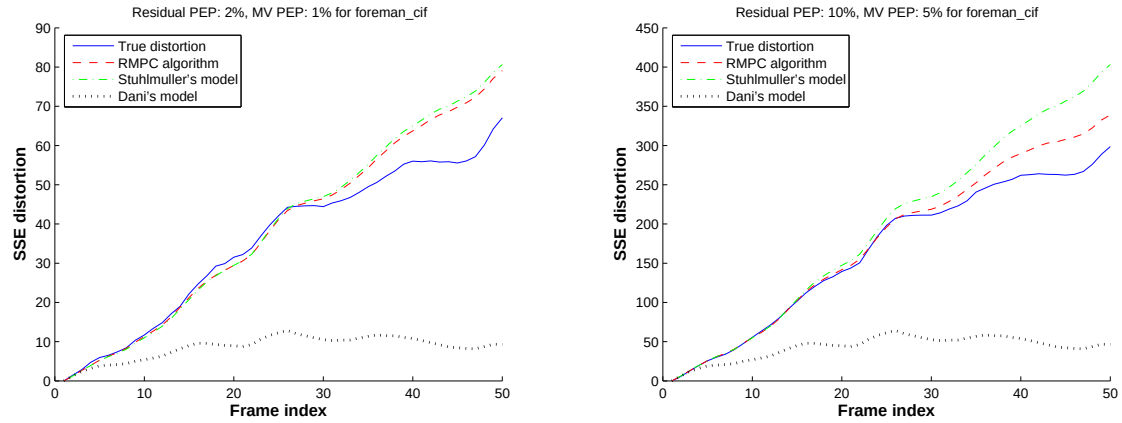
the correlation caused distortion is small for low motion sequence under low PEP as proved in Ref. [1], linear model is quite accurate as shown in Fig. 3(a). Notice that in foreman sequence under good channel, the estimated distortion different from ground truth is only about $MSE = 12$ after accumulated with 50 frames without feedback.

In Ref. [3], authors claim that the larger the fraction of pixels in the reference frame to be used as reference pixels, the larger the transmission errors propagated from the reference frame. However, due to randomness of motion vectors, the probability that a pixel with error is used as reference is the same as the probability that a pixel without error is used as reference. Therefore, the number of pixels in the reference frame being used for motion prediction has nothing to do with the fading factor. As a result, the algorithm in Ref. [3] under-estimates transmission distortion as shown in Fig. 2 and Fig. 3.



(a) (b)

Figure 2: Transmission distortion D^k vs. frame index k for 'stefan': (a) good channel, (b) poor channel.



(a) (b)

Figure 3: Transmission distortion D^k vs. frame index k for 'foreman': (a) good channel, (b) poor channel.

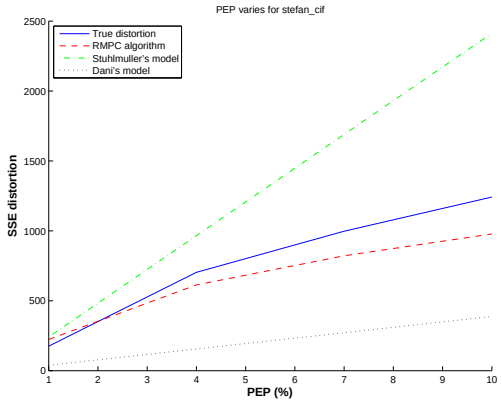


Figure 4: Transmission distortion D^k vs. PEP for 'stefan'.

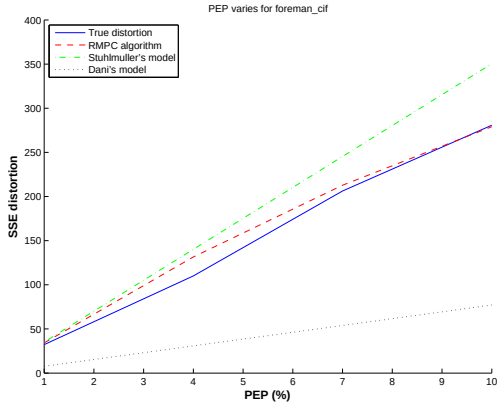


Figure 5: Transmission distortion D^k vs. PEP for 'foreman'.

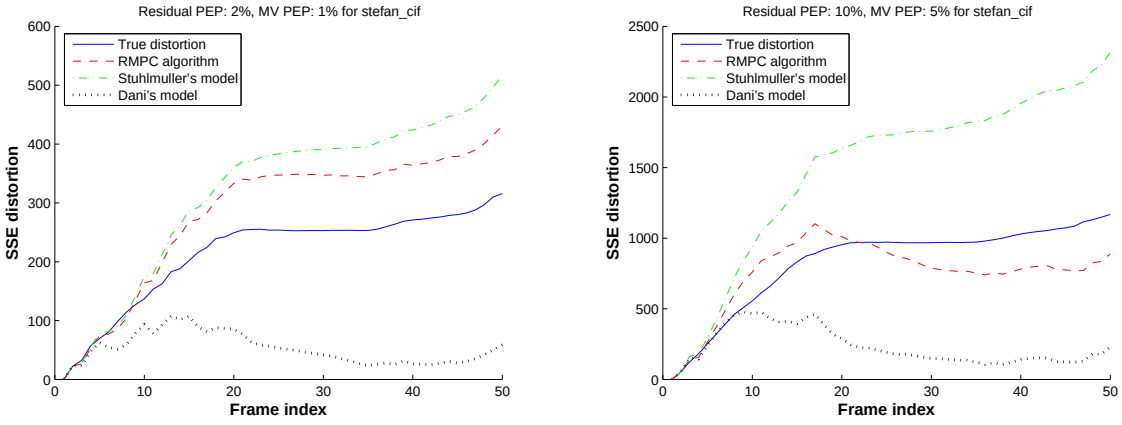
In our experiment, we observe that 1) the higher the propagated distortion, the smaller the propagation factor; and 2) the higher percentage of reconstructed pixel values near 0 or 255, the smaller the propagation factor. These two phenomena once more verify that the propagation factor is a function of all samples of reconstructed pixel value and sample variance of propagated error as proved in Ref. [1]. These phenomena could be explained by (13) in that 1) α is a decreasing function of b for $b > 0$; 2) α is an increasing function of y for $0 \leq y \leq 127$ and a decreasing function of y for $128 \leq y \leq 255$. We also note that a larger sample variance of propagated error causes α to be less sensitive to the change of reconstructed pixel value, while a larger deviation of reconstructed pixel value from 128 causes α to be less sensitive to the change of sample variance of propagated error.

To further study estimation accuracy, we test the estimation algorithms under many different channel conditions. Fig. 4 and Fig. 5 show the estimation accuracy under PEP varying from 1% to 10%. In both figures, RMPC-FTD algorithm achieves the most accurate distortion estimation under all channel conditions.

7.1.3. Robustness of Different Estimation Algorithms

In Section 7.1.2, we assume PEP is perfectly known at the encoder. However, in a real wireless video communication system, PEP is usually not perfectly known at the encoder; i.e., there is a random estimation error between the true PEP and the estimated PEP. Hence, it is important to evaluate the robustness of the estimation algorithms against PEP estimation error. To simulate imperfect PEP estimation, for a given true PEP denoted by P_{true}, we assume the estimated PEP is a random variable and is uniformly distributed in $[0, 2 \times P_{true}]$; e.g., if $P_{true} = 10\%$, the estimated PEP is uniformly distributed in $[0, 20\%]$.

Figs. 6 and 7 show the estimation accuracy of the three algorithms for 'stefan' and 'foreman', respectively, under imperfect knowledge of PEP at the encoder. From the two figures, it is observed that compared to the case under perfect knowledge of PEP at the encoder, for both Stuhlmuller's model and Dani's model, imperfect knowledge of PEP may cause increase or decrease of the gap between the estimated distortion and the true distortion. Specifically, for Stuhlmuller's model, if the PEP is under-estimated, the gap between the estimated distortion and the true distortion decreases, compared to the case under perfect knowledge of PEP; for Dani's model, if the PEP is over-estimated, the gap between the estimated distortion and the true distortion decreases, compared to the case under perfect knowledge of PEP. In contrast, RMPC-FTD algorithm is more robust against PEP estimation error, and provides more accurate distortion estimate than Stuhlmuller's model and Dani's model.

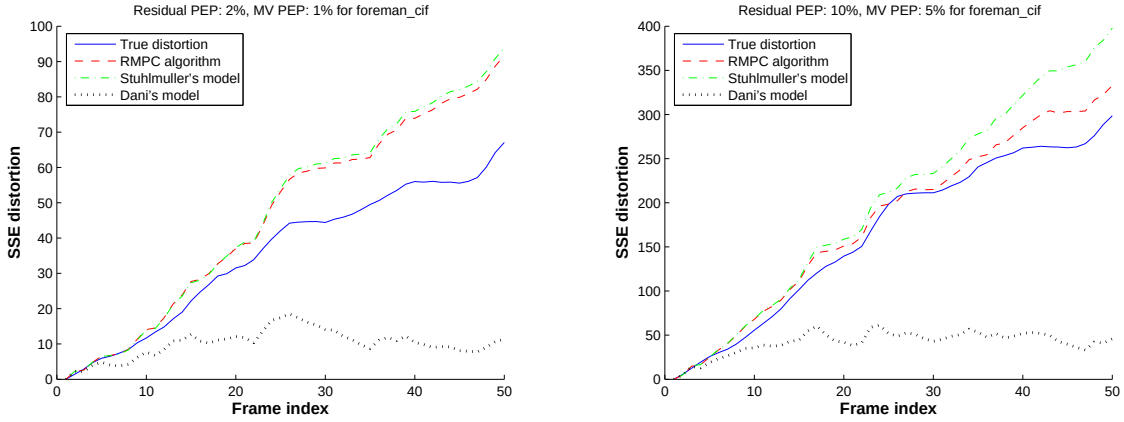


(a) (b)

Figure 6: Transmission distortion D^k vs. frame index k for 'stefan' under imperfect knowledge of PEP: (a) good channel, (b) poor channel.

7.2. R-D Performance of Mode Decision Algorithms

In this subsection, we compare the R-D performance of Algorithm 2 with that of ROPE and LLN algorithms for mode decision in H.264. To compare all algorithms under the multi-reference picture motion compensated prediction, we also enhance the original ROPE algorithm [8] with multi-reference capability.



(a) (b)

Figure 7: Transmission distortion D^k vs. frame index k for 'foreman' under imperfect knowledge of PEP: (a) good channel, (b) poor channel.

7.2.1. Experiment Setup

All three algorithms are integrated into JM16.0 [15] encoder. To support more advanced techniques in H.264, we implement them in the high profile defined in H.264 specification Annex A [14]. We then conduct experiments for five schemes, that is, three ERRDO schemes (RMPC-MS, LLN, ROPE), random intra update, and default SCRDO scheme with no transmission distortion estimation. All the tested video sequences are in CIF resolution with 30fps. Each coded video sequence is tested under a variety of PEP settings from 0.5% to 5%. Each video sequence is coded for its first 100 frames with 3 slices per frame. The error concealment method used in the experiment is to copy the pixel value in the same position of the previous frame. For all algorithms, the first frame is assumed to be correctly received.

The encoder setting is given as below. No slice data partitioning is used; constrained intra prediction is enabled; the number of reference frames is 3; B-MBs are not included; only 4x4 transform is used; CABAC is enabled for entropy coding; in LLN algorithm, the number of simulated decoders is 30.

7.2.2. R-D performance under no interpolation filter and no deblocking filter

In the experiments of this subsection, both the deblocking filter and the interpolation filter with fractional MV in H.264 are disabled. Due to the space limit, we only show the plot of PSNR vs. bit rate for video sequences 'foreman' and 'football' under $PEP = 2\%$ and $PEP = 5\%$, with rate control enabled. Figs. 8 and 9 show PSNR vs. bit rate for 'foreman' and 'football', respectively. The experimental results show that RMPC-MS achieves the best R-D performance; LLN and ROPE achieves similar performance and the second best R-D performance; the random intra update scheme (denoted by 'RANDOM') achieves the third best R-D performance; the SCRDO scheme (denoted by 'NO_EST') achieves the worst R-D performance.

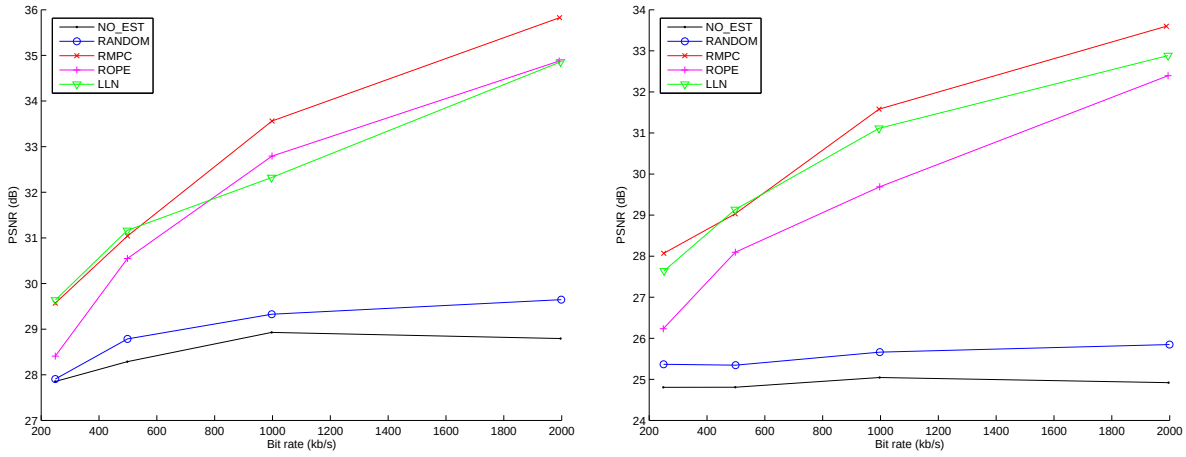


Figure 8: PSNR vs. bit rate for 'foreman', with no interpolation filter and no deblocking filter: (a) PEP=2%, (b) PEP=5%.

LLN has poorer R-D performance than RMPC-MS; this may be because 30 simulated decoders are still not enough to achieve reliable distortion estimate although LLN with 30 simulated decoders already incurs much higher complexity than RMPC-MS. The reason why RMPC-MS achieves better R-D performance than ROPE, is due to the consideration of clipping noise in RMPC-MS. Debug messages show that, without considering the clipping noise, ROPE over-estimates the end-to-end distortion for inter modes; hence ROPE tends to select intra modes more often than RMPC-MS and LLN, which leads to higher encoding bit rate in ROPE; as a result, the PSNR gain achieved by ROPE is compromised by its higher bit rate. To verify this conjecture, we test all sequences under the same Quantization Parameter (QP) settings and without rate control. we observe that ROPE algorithm always produces higher bit rate than other schemes.

Table 2 shows the average PSNR gain (in dB) of RMPC-MS over ROPE and LLN for different video sequences and different PEP. The average PSNR gain is obtained by the method in Ref. [16], which measures average distance (in PSNR) between two R-D curves. From Table 2, we see that RMPC-MS achieves an average PSNR gain of 1.44dB over ROPE for 'foreman' under $PER = 5\%$; and it achieves an average PSNR gain of 0.89dB over LLN for 'foreman' sequence under $PEP = 1\%$.

Table 2: Average PSNR gain (in dB) of RMPC-MS over ROPE and LLN

Sequence	coastguard				football				foreman				mobile			
PEP	5%	2%	1%	0.5%	5%	2%	1%	0.5%	5%	2%	1%	0.5%	5%	2%	1%	0.5%
RMPC vs. ROPE	0.75	0.26	0.15	0.16	0.88	0.26	0.19	0.22	**1.44**	0.74	0.61	0.30	0.51	0.14	0.15	0.10
RMPC vs. LLN	0.04	0.23	0.20	0.17	0.42	0.22	0.28	0.15	0.28	0.53	**0.89**	0.65	0.29	0.23	0.28	0.19

7.2.3. R-D performance with interpolation filter and deblocking filter

In H.264, interpolation filter provides notable objective (PSNR) gain and deblocking filter provides notable subjective gain. To support the interpolation filter with fractional

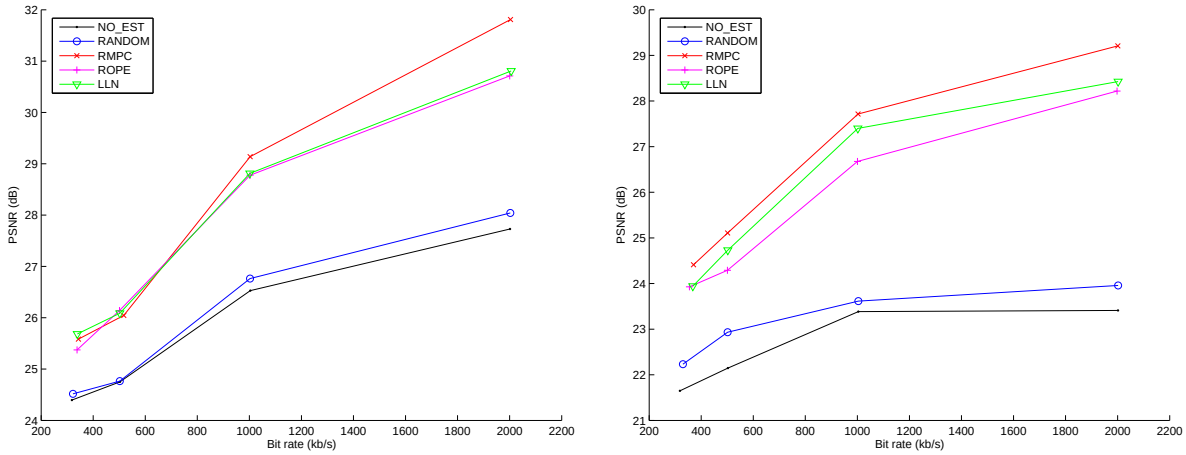


Figure 9: PSNR vs. bit rate for 'football', with no interpolation filter and no deblocking filter: (a) PEP=2%, (b) PEP=5%.

MV in H.264 [17], we extend Algorithm 2 by using the nearest neighbor to approximate the reference pixel pointed by a fractional MV. In addition, deblocking filter is also enabled in JM16.0 to compare RMPC-MS, ROPE and LLN algorithms.

Note that both RMPC-MS and ROPE are derived without considering filtering operations. Due to high spatial correlation between adjacent pixels, the averaging operation induced by a filter will produce many cross-correlation terms for estimating distortion in a subpixel position. Yang et al. [18] enhance the original ROPE algorithm with interpolation filter in H.264. However, their algorithm requires 1 square root operation, 1 exponentiation operation, and 2 multiplication operations for calculating each cross-correlation term. Since a six-tap interpolation filter is used in H.264 for subpixel accuracy of motion vector, there are 15 cross-correlation terms for calculating each subpixel distortion. Therefore, the complexity of their algorithm is very high and may not be suitable for real-time encoding. In this subsection, we use a very simple but R-D efficient method to estimate subpixel distortion. Specifically, we choose the nearest integer pixel around the subpixel, and use the distortion of the nearest integer pixel as the estimated distortion for the subpixel. Note that this simple method is not aimed at extending RMPC-MS and ROPE algorithms, but just to compare the R-D performances of these two algorithms for H.264 with fractional MV for motion compensation.

We first show the experiment results with interpolation filter but with no deblocking filter as in Figs. 10 and 11. From Figs. 10 and 11, we observe the same result as shown in Section 7.2.2: RMPC-MS achieves better R-D performance than LLN and ROPE algorithms. From Figs. 10 and 11, we also can see that each of the five algorithms achieves higher PSNR than its corresponding scheme with no interpolation filter; this means the simple method is valid. We also observe from Table 3 that in this case, RMPC-MS achieves an average PSNR gain of 2.97dB over ROPE for sequence 'mobile' under $PEP = 0.5\%$; and it achieves an average PSNR gain of 1.13dB over LLN for 'foreman' under $PEP = 1\%$.

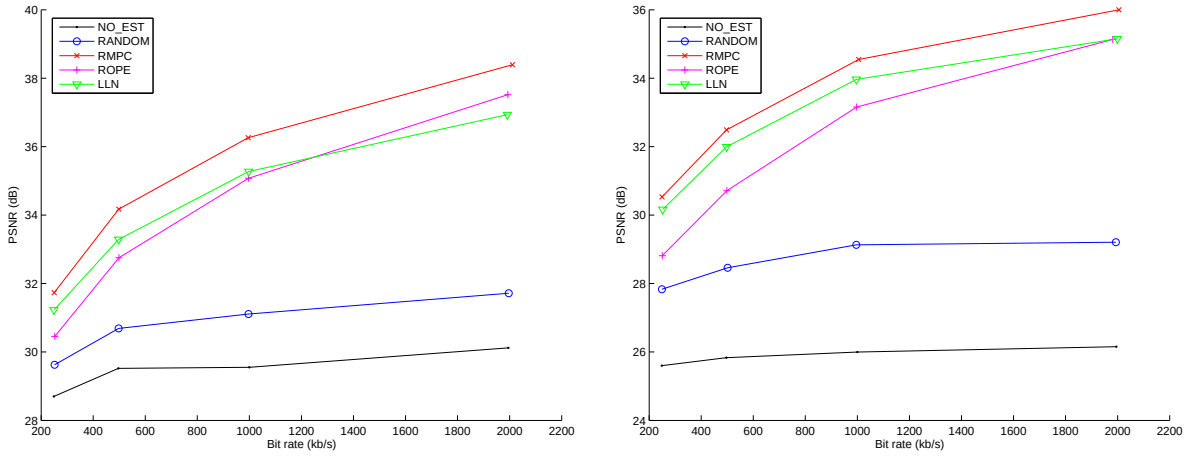


Figure 10: PSNR vs. bit rate for 'foreman', with interpolation and no deblocking: (a) PEP=2%, (b) PEP=5%.

Table 3: Average PSNR gain (in dB) of RMPC-MS over ROPE and LLN under interpolation filtering

Sequence	coastguard				football				foreman				mobile			
PEP	5%	2%	1%	0.5%	5%	2%	1%	0.5%	5%	2%	1%	0.5%	5%	2%	1%	0.5%
RMPC vs. ROPE	0.49	0.38	0.43	0.56	0.45	0.24	0.25	0.30	1.51	1.25	1.20	1.25	0.92	1.64	2.58	**2.97**
RMPC vs. LLN	0.23	0.28	0.38	0.31	0.27	0.38	0.35	0.23	0.56	0.95	**1.13**	1.07	0.30	0.57	0.35	0.33

We also show the experiment results with both interpolation filter and deblocking filter as shown in Figs. 12 and 13. It is interesting to see that each of the five algorithms with interpolation filter and deblocking filter achieves poorer R-D performance than the corresponding one with interpolation filter and no deblocking filter. That is, adding deblocking filter degrades the R-D performance of each algorithm since their estimated distortions become less accurate. In this case, ROPE sometimes performs better than RMPC-MS; this can be seen in Fig. 13, which is also the only case we have observed that ROPE performs better than RMPC-MS. This may be because RMPC-MS has a higher percentage of inter modes than ROPE. Since the deblocking operation is executed after the error concealment as in JM16.0, for intra prediction, deblocking filter only affects the estimated distortion if the packet is lost; for inter prediction, deblocking filter always impacts the estimated distortion. Therefore, the estimation accuracy for inter prediction suffers from deblocking filter more than that for intra prediction. Thus, it is likely that more inter modes in RMPC-MS cause higher PSNR drop in Fig. 13.

8. Conclusion

In this paper, we designed RMPC-FTD, RMPC-PTD, RMPC-PEED algorithms based on the analysis in Part I [1]. By virtue of considering the non-linear clipping noise and the negative correlation between the MV concealment error and the propagated error, RMPC-FTD algorithm provides more accurate FTD estimation than existing models as verified by

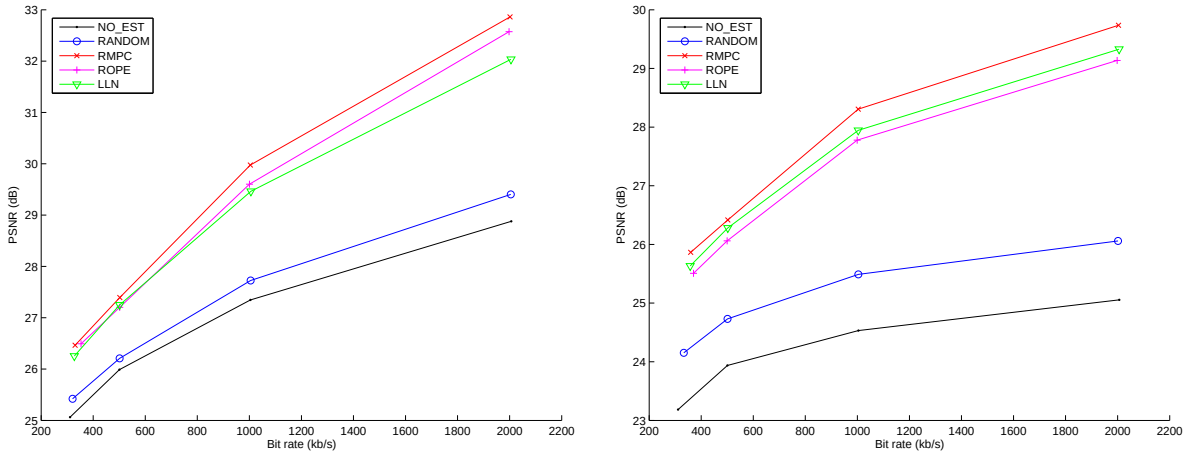


Figure 11: PSNR vs. bit rate for 'football', with interpolation and no deblocking: (a) PEP=2%, (b) PEP=5%.

experimental results. In addition, experimental results also show that RMPC-FTD algorithm is more robust against inaccurate estimation of PEP than existing models. We also designed RMPC-MS algorithm for mode decision in H.264 and enhance it with support of interpolation filter and deblocking filter. Experimental results show that our RMPC-MS algorithm achieves an average PSNR gain of 1.44dB over ROPE algorithm for 'foreman' sequence under $PEP = 5\%$; and it achieves an average PSNR gain of 0.89dB over LLN algorithm for 'foreman' sequence under $PEP = 1\%$.

There are three directions for our future work. First, in Section 7.2.3, we enhanced RMPC-MS algorithm with support of interpolation filter by a simple method. The method is just an approximation by the nearest neighbor integer pixel. Therefore, the resulting R-D performance is not optimal although it is still better than LLN and ROPE algorithm. To further improve the R-D performance of RMPC-MS, we will theoretically derive a new distortion estimation algorithm based on RMPC-MS algorithm to support averaging operations, such as interpolation, deblocking, and B-MB.

Second, we will continue our work on the ERRDO problem. Current ERRDO mode decision in JM16.0 still uses the same Lagrange multiplier λ as that for source coding RDO. However, in an error-prone channel, λ is a function of video content, MV, mode, QP, PEP, error concealment scheme, and constrained bit rate. Our future research topic is to analytically derive the optimal λ for wireless video transmission, and then design an ERRDO scheme for joint MV, mode, QP selection.

Third, we will extend our current work to cross layer rate control. The rate control algorithms in existing video standards, for example, H.263 [19] and H.264 [20, 21] only focus on source-coding rate control but not cross layer rate control. Specifically, existing rate control algorithms only optimize R-D cost over QP, which determines the source-coding bit rate. However, as we see in this paper, PEP is another important parameter that affects the end-to-end distortion; and PEP determines the number of redundant bits resulted from

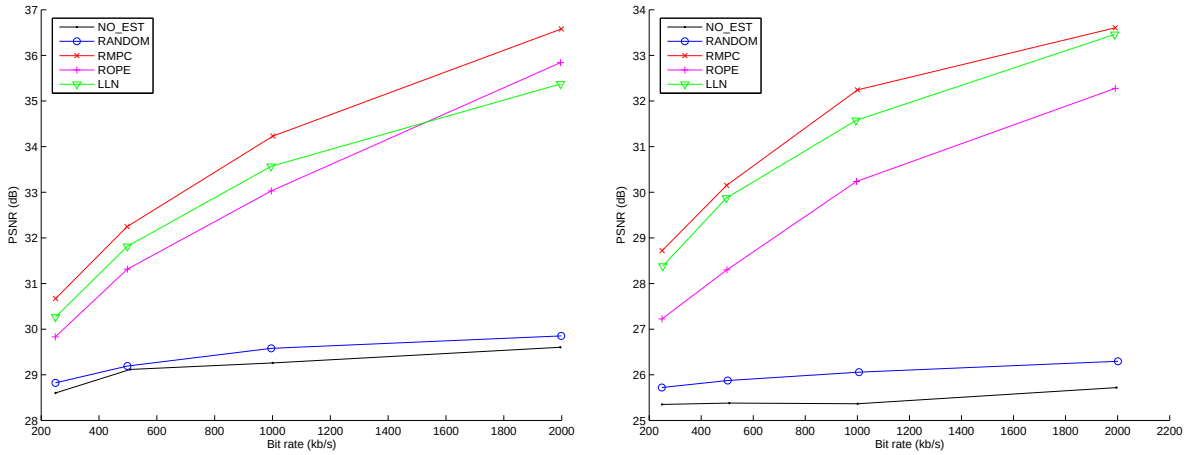


Figure 12: PSNR vs. bit rate for 'foreman', with interpolation and deblocking: (a) PEP=2%, (b) PEP=5%.

channel coding. In our future work, we will study optimal bit allocation for source coding and channel coding that minimizes the end-to-end distortion; the tuning parameters include QP and physical layer parameters such as channel coding rate and power.

Acknowledgments

This work was supported in part by an Intel gift, the US National Science Foundation under grant DBI-0529012 and CNS-0643731. The authors would like to thank Peshala Pahalawatta and Alexis Michael Tourapis for many fruitful discussions related to this work.

Appendix A. Derivation of Transmission Distortion Formulae

Appendix A.1. Definitions

The reconstructed pixel value for $\mathbf{u}^k$ at the encoder is

$$\hat{f}_{\mathbf{u}}^k = \Gamma(\hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} + \hat{e}_{\mathbf{u}}^k), \tag{A.1}$$

where $\Gamma(\cdot)$ function is a clipping function defined by

$$\Gamma(x) = \begin{cases} \gamma_L, & x < \gamma_L \\ x, & \gamma_L \le x \le \gamma_H \\ \gamma_H, & x > \gamma_H, \end{cases} \tag{A.2}$$

where γ_L and γ_H are user-specified low threshold and high threshold, respectively. Usually, $\gamma_L = 0$ and $\gamma_H = 255$.

The reconstructed pixel value for $\mathbf{u}^k$ at the decoder is

$$\widetilde{f}_{\mathbf{u}}^k = \Gamma(\widetilde{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{e}_{\mathbf{u}}^k). \tag{A.3}$$

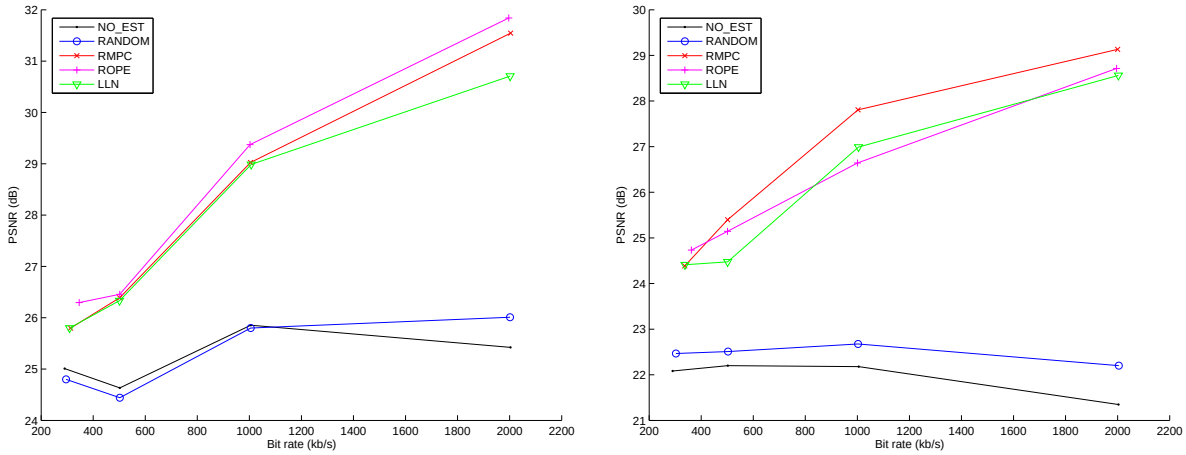


Figure 13: PSNR vs. bit rate for 'football', with interpolation and deblocking: (a) PEP=2%, (b) PEP=5%.

We define the transmission distortion for pixel $\mathbf{u}^k$ or PTD by

$$D_{\mathbf{u}}^k \triangleq E[(\hat{f}_{\mathbf{u}}^k - \widetilde{f}_{\mathbf{u}}^k)^2], \tag{A.4}$$

and we define the transmission distortion for the k-th frame or FTD by

$$D^k \triangleq E[\frac{1}{|\mathcal{V}|} \cdot \sum_{\mathbf{u}\in\mathcal{V}} (\hat{f}_{\mathbf{u}}^k - \widetilde{f}_{\mathbf{u}}^k)^2]. \tag{A.5}$$

It is easy to prove that the relationship between FTD and PTD is characterized by

$$D^k = \frac{1}{|\mathcal{V}|} \cdot \sum_{\mathbf{u}\in\mathcal{V}} D_{\mathbf{u}}^k. \tag{A.6}$$

We define the clipping noise for pixel $\mathbf{u}^k$ at the encoder as

$$\hat{\Delta}_{\mathbf{u}}^k \triangleq (\hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} + \hat{e}_{\mathbf{u}}^k) - \Gamma(\hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} + \hat{e}_{\mathbf{u}}^k), \tag{A.7}$$

and the clipping noise for pixel $\mathbf{u}^k$ at the decoder as

$$\widetilde{\Delta}_{\mathbf{u}}^k \triangleq (\widetilde{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{e}_{\mathbf{u}}^k) - \Gamma(\widetilde{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{e}_{\mathbf{u}}^k). \tag{A.8}$$

Using (A.1), Eq. (A.7) becomes

$$\hat{f}_{\mathbf{u}}^k = \hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} + \hat{e}_{\mathbf{u}}^k - \hat{\Delta}_{\mathbf{u}}^k, \tag{A.9}$$

and using (A.3), Eq. (A.8) becomes

$$\widetilde{f}_{\mathbf{u}}^k = \widetilde{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{e}_{\mathbf{u}}^k - \widetilde{\Delta}_{\mathbf{u}}^k, \tag{A.10}$$

Appendix A.2. Overview of the Approach to Analyzing PTD and FTD

To analyze PTD and FTD, we take a divide-and-conquer approach. We first divide transmission reconstructed error into four components: three random errors (RCE, MVCE and propagated error) due to their different physical causes, and clipping noise, which is a non-linear function of these three random errors. This error decomposition allows us to further decompose transmission distortion into four terms, i.e., distortion caused by 1) RCE, 2) MVCE, 3) propagated error plus clipping noise, and 4) correlations between any two of the error sources, respectively. This distortion decomposition facilitates the derivation of a simple and accurate closed-form formula for each of the four distortion terms. Next, we elaborate on error decomposition and distortion decomposition.

Define transmission reconstructed error for pixel $\mathbf{u}^k$ by $\widetilde{\zeta}_{\mathbf{u}}^k \triangleq \hat{f}_{\mathbf{u}}^k - \widetilde{f}_{\mathbf{u}}^k$. From (A.9) and (A.10), we obtain

$$\begin{aligned}\widetilde{\zeta}_{\mathbf{u}}^k &= (\hat{e}_{\mathbf{u}}^k + \hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} - \hat{\Delta}_{\mathbf{u}}^k) - (\widetilde{e}_{\mathbf{u}}^k + \widetilde{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} - \widetilde{\Delta}_{\mathbf{u}}^k) \\ &= (\hat{e}_{\mathbf{u}}^k - \widetilde{e}_{\mathbf{u}}^k) + (\hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} - \hat{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}) + (\hat{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} - \widetilde{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}) - (\hat{\Delta}_{\mathbf{u}}^k - \widetilde{\Delta}_{\mathbf{u}}^k).\end{aligned} \tag{A.11}$$

Define RCE $\widetilde{\varepsilon}_{\mathbf{u}}^k$ by $\widetilde{\varepsilon}_{\mathbf{u}}^k \triangleq \hat{e}_{\mathbf{u}}^k - \widetilde{e}_{\mathbf{u}}^k$, and define MVCE $\widetilde{\xi}_{\mathbf{u}}^k$ by $\widetilde{\xi}_{\mathbf{u}}^k \triangleq \hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-1} - \hat{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}$. Note that $\hat{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} - \widetilde{f}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} = \widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}$, which is the transmission reconstructed error of the concealed reference pixel in the reference frame; we call $\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1}$ propagated error. As mentioned in Ref. [1], we assume $\hat{\Delta}_{\mathbf{u}}^k = 0$. Therefore, (A.11) becomes

$$\widetilde{\zeta}_{\mathbf{u}}^k = \widetilde{\varepsilon}_{\mathbf{u}}^k + \widetilde{\xi}_{\mathbf{u}}^k + \widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k. \tag{A.12}$$

(A.12) is our proposed **error decomposition**.

Combining (A.4) and (A.12), we have

$$\begin{aligned}D_{\mathbf{u}}^k &= E[(\widetilde{\varepsilon}_{\mathbf{u}}^k + \widetilde{\xi}_{\mathbf{u}}^k + \widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k)^2] \\ &= E[(\widetilde{\varepsilon}_{\mathbf{u}}^k)^2] + E[(\widetilde{\xi}_{\mathbf{u}}^k)^2] + E[(\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k)^2] \\ &+ 2E[\widetilde{\varepsilon}_{\mathbf{u}}^k \cdot \widetilde{\xi}_{\mathbf{u}}^k] + 2E[\widetilde{\varepsilon}_{\mathbf{u}}^k \cdot (\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k)] + 2E[\widetilde{\xi}_{\mathbf{u}}^k \cdot (\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k)].\end{aligned} \tag{A.13}$$

Denote $D_{\mathbf{u}}^k(r) \triangleq E[(\widetilde{\varepsilon}_{\mathbf{u}}^k)^2]$, $D_{\mathbf{u}}^k(m) \triangleq E[(\widetilde{\xi}_{\mathbf{u}}^k)^2]$, $D_{\mathbf{u}}^k(P) \triangleq E[(\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k)^2]$ and $D_{\mathbf{u}}^k(c) \triangleq 2E[\widetilde{\varepsilon}_{\mathbf{u}}^k \cdot \widetilde{\xi}_{\mathbf{u}}^k] + 2E[\widetilde{\varepsilon}_{\mathbf{u}}^k \cdot (\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k)] + 2E[\widetilde{\xi}_{\mathbf{u}}^k \cdot (\widetilde{\zeta}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k}^{k-1} + \widetilde{\Delta}_{\mathbf{u}}^k)]$. Then, (A.13) becomes (6), which is our proposed **distortion decomposition** for PTD. The reason why we combine propagated error and clipping noise into one term (called clipped propagated error) is because clipping noise is mainly caused by propagated error and such decomposition will simplify the formulae.

To derive the formula for FTD, from (A.6) and (6), we obtain (1), where

$$D^k(r) = \frac{1}{|\mathcal{V}|} \cdot \sum_{\mathbf{u} \in \mathcal{V}} D_{\mathbf{u}}^k(r), \tag{A.14}$$

$$D^k(m) = \frac{1}{|\mathcal{V}|} \cdot \sum_{\mathbf{u}\in\mathcal{V}} D^k_{\mathbf{u}}(m), \tag{A.15}$$

$$D^k(P) = \frac{1}{|\mathcal{V}|} \cdot \sum_{\mathbf{u}\in\mathcal{V}} D^k_{\mathbf{u}}(P), \tag{A.16}$$

$$D^k(c) = \frac{1}{|\mathcal{V}|} \cdot \sum_{\mathbf{u}\in\mathcal{V}} D^k_{\mathbf{u}}(c), \tag{A.17}$$

which is our proposed distortion decomposition for FTD.

Following the detailed derivation process in Ref. [1], we have (1)-(10).

Appendix B. Variables for the Estimation of Residual Caused Distortion

Concealment of $\hat{e}^k_{\mathbf{u}}$ at the decoder: At the decoder, if $\hat{e}^k_{\mathbf{u}}$ is received with error and its neighboring pixels are correctly received, its neighboring pixels could be utilized to conceal $\hat{e}^k_{\mathbf{u}}$. However, this is possible only if the pixel $\mathbf{u}^k$ is at the slice boundary and the pixels at the other side of this slice boundary is correctly received. In H.264, most pixels in a slice do not locate at the slice boundary. Therefore, if one slice is lost, most of pixels in that slice will be concealed without the information from neighboring pixels. If the same method is used to conceal $\check{e}^k_{\mathbf{u}}$ of all pixels, it is not difficult to prove that the minimum of $E[(\varepsilon^k)^2]$ is achieved when $\check{e}^k_{\mathbf{u}} = E[\hat{e}^k_{\mathbf{u}}]$.

Note that when $\check{e}^k_{\mathbf{u}}$ is concealed by $E[\hat{e}^k_{\mathbf{u}}]$ at the decoder, $E[(\varepsilon^k)^2]$ is the variance of $\hat{e}^k_{\mathbf{u}}$, that is, $E[(\varepsilon^k)^2] = \sigma^2_{\hat{e}^k_{\mathbf{u}}}$. In our experiment, we find that the histogram of $\hat{e}^k$ in each frame approximately follows a Laplacian distribution with zero mean. As proved in Ref. [22], the variance of e^k depends on the spatio-temporal correlation of the input video sequence and the accuracy of motion estimation. Since $\hat{e}^k$ is a function of e^k, $E[(\varepsilon^k)^2]$ also depends on the accuracy of motion estimation. So, for a given video sequence, more accurate residual concealment and more accurate motion estimation produce a smaller $D^k(r)$. This could be used as a criterion for the design of the encoding algorithm at the encoder and residual concealment method at the decoder.

Estimation of $\check{e}^k_{\mathbf{u}}$ at the encoder: If the encoder has knowledge of the concealment method used by the decoder as well as the feedback acknowledgement of some packets, $\check{e}^k_{\mathbf{u}}$ can be estimated by the same concealed methods used by the decoder. That means the methods to estimate $\check{e}^k_{\mathbf{u}}$ of pixels at the slice boundary are different from other pixels. However, if no feedback acknowledgement of which packets are correctly received, the same method may be used to estimate $\check{e}^k_{\mathbf{u}}$ of all pixels, that is, $\widehat{\check{e}^k_{\mathbf{u}}} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k} \hat{e}^k_{\mathbf{u}}$. Note that even if the feedback acknowledgement of some packets are correctly received before the estimation, the estimate obtained by this method at the encoder is still quite accurate since most pixels in a slice do not locate at the slice boundary.

In most cases, for a standard hybrid codec such as H.264, $\frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k} \hat{e}^k_{\mathbf{u}}$ approximately equals zero[6] for P-MBs and B-MBs. Therefore, one simple concealment method is to let

[6]This is actually an objective of predictive coding.

$\widehat{\check{e}}^k_{\mathbf{u}} = 0$ as in most transmission distortion models. In this paper, we still use $\widehat{\check{e}}^k_{\mathbf{u}}$ in case $\frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\hat{e}^k_{\mathbf{u}} \neq 0$ due to the imperfect predictive coding, or in the general case, that is, some feedback acknowledgements may have been received before the estimation. Note that when $\widehat{\check{e}}^k_{\mathbf{u}} = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{u}\in\mathcal{V}^k}\hat{e}^k_{\mathbf{u}}$ at the encoder, $\hat{E}[(\varepsilon^k)^2]$ is the sample variance of $\hat{e}^k_{\mathbf{u}}$ and in fact a biased estimator of $\sigma^2_{\hat{e}^k_{\mathbf{u}}}$ [23]. In other words, $\hat{E}[(\varepsilon^k)^2]$ is a sufficient statistic of all individual samples $\hat{e}^k_{\mathbf{u}}$. If the sufficient statistic $\hat{E}[(\varepsilon^k)^2]$ is known, the FTD estimator does not need the values of $\hat{e}^k_{\mathbf{u}}$ of all pixels. Therefore, such an FTD estimator incurs much lower complexity than using the values of $\hat{e}^k_{\mathbf{u}}$ of all pixels.

Estimation of $P^k_i(r)$: In wired communication, application layer PEP is usually estimated by Packet Error Rate (PER), which is the ratio of the number of incorrectly received packets to the number of transmitted packets, that is, $\hat{P}^k_i(r) = PER^k_i(r)$. In a wireless fading channel, instantaneous physical layer PEP is a function of the instantaneous channel gain $g(t)$ at time t [24], which is denoted by $p(g(t))$. At an encoder, there are two cases: 1) the transmitter has perfect knowledge of $g(t)$, and 2) the transmitter has no knowledge of $g(t)$ but knows the probability density function (pdf) of $g(t)$. For Case 1, the estimated PEP $\hat{P}^k_i(r) = p(g(t))$ since $g(t)$ is known. Note that since the channel gain is time varying, the estimated instantaneous PEP is also time varying.[7] For Case 2, $p(g(t))$ is a random variable since only pdf of $g(t)$ is known. Hence, we should use the expected value of $p(g(t))$ to estimate $P^k_i(r)$, that is, $\hat{P}^k_i(r) = E[p(g(t))]$, where the expectation is taken over the pdf of $g(t)$.

Appendix C. Variables for the Estimation of MV Caused Distortion

Concealment of $\mathbf{mv}^k_{\mathbf{u}}$ at the decoder: Different from residual, MV are highly correlated in both temporal and spatial domains. Hence, the decoder may conceal the MV by temporally neighboring block if its spatially neighboring blocks are not available. Depending on whether the neighboring blocks are correctly received or not, there may be several options of MV error concealment methods for each block, or each pixel to make it more general. If the neighboring blocks are correctly received, $\check{\mathbf{mv}}^k_{\mathbf{u}}$ can be concealed by the median or average of those neighboring blocks. Interested readers may refer to Refs. [25, 26, 27] for discussions on different MV concealment methods. In our experiment, we also observe that the histogram of ξ^k in one frame approximately follows a Laplacian distribution with zero mean. For different concealment methods, the variance of ξ^k will be different. Therefore, the more accurate concealed motion estimation, the smaller $D^k(m)$.

Estimation of $\check{\mathbf{mv}}^k_{\mathbf{u}}$ at the encoder: If the encoder knows the concealment methods of current block and the PEP of neighboring blocks, we can estimate the MV caused distortion by assigning different concealment methods with different probabilities at the encoder as in Ref. [8]. However, if the encoder does not know what concealment methods are used by the decoder or no neighboring blocks can be utilized for error concealment (e.g., both temporal and spatial neighboring blocks are in error), a simple estimation algorithm [3], [4] is to let

[7] This implies that the pixel error process is non-stationary over both time and space [1].

$\widehat{\check{\mathbf{mv}}_{\mathbf{u}}^k} = 0$, that is, using the pixel value from the same position of the previous frame. In this paper, we still use $\widehat{\check{\mathbf{mv}}_{\mathbf{u}}^k}$ to denote the estimate of concealed motion vector for the general case.

Estimation of $P_i^k(m)$: The estimation of $P_i^k(m)$ is similar to the estimation of $P_i^k(r)$. Note that in H.264 specification, there is no slice data partitioning for an instantaneous decoding refresh (IDR) frame [14], so $P_i^k(r) = P_i^k(m)$ for all pixels in an IDR-frame. This is also true for I-MB, and P-MB without slice data partitioning. For P-MB with slice data partitioning in H.264, the error state of residual and the error state of MV of the same pixel are partially correlated. To be more specific, if the MV packet is lost, the corresponding residual packet cannot be decoded even if it is correctly received, since there is no slice header in the residual packet. As a result, $P_i^k(r_{H.264}) = P_i^k(r) + (1 - P_i^k(r))P_i^k(m)$.

Appendix D. Proof of Proposition 1

Proof 1. *From Ref. [1], we know that* $\widetilde{\Delta}_{\mathbf{u}}^k \triangleq (\widetilde{f}^{k-j'}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k} + \widetilde{e}_{\mathbf{u}}^k) - \Gamma(\widetilde{f}^{k-j'}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k} + \widetilde{e}_{\mathbf{u}}^k)$. *If there is no newly induced error, that is,* $\widetilde{e}_{\mathbf{u}}^k = \hat{e}_{\mathbf{u}}^k$ *and* $\widetilde{\mathbf{mv}}_{\mathbf{u}}^k = \mathbf{mv}_{\mathbf{u}}^k$, *we have* $\widetilde{f}^{k-j'}_{\mathbf{u}+\widetilde{\mathbf{mv}}_{\mathbf{u}}^k} + \widetilde{e}_{\mathbf{u}}^k = \widetilde{f}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k} + \hat{e}_{\mathbf{u}}^k = \hat{f}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k} - \widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k} + \hat{e}_{\mathbf{u}}^k = \hat{f}_{\mathbf{u}}^k - \widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}$. *Therefore, we have*

$$\widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r},\bar{m}\} = \begin{cases} \hat{f}_{\mathbf{u}}^k - \widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k} - 255, & \hat{f}_{\mathbf{u}}^k - \widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k} > 255 \\ \hat{f}_{\mathbf{u}}^k - \widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}, & \hat{f}_{\mathbf{u}}^k - \widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k} < 0 \\ 0, & \textit{otherwise.} \end{cases} \tag{D.1}$$

Adding $\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}$ *to the left-hand side and right-hand side in (D.1), we obtain (23).*

Appendix E. Proof of Theorem 1

Proof 2. *Since there is no slice data partitioning,* $D_{\mathbf{u},ETE}^k = (1 - P_{\mathbf{u}}^k) \cdot D_{\mathbf{u},ETE}^k\{\bar{r},\bar{m}\} + P_{\mathbf{u}}^k \cdot D_{\mathbf{u},ETE}^k\{r,m\}$.

First, if the packet is lost, from (20) we obtain

$$\begin{aligned} D_{\mathbf{u}}^k\{r,m\} &= (\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k)^2 + 2(\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k) \cdot E[\widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}] + D^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k} \\ &= (\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k + E[\widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}])^2 + \sigma^2_{\widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}}, \end{aligned} \tag{E.1}$$

and from (21) we obtain

$$E[\widetilde{\zeta}_{\mathbf{u}}^k]\{r,m\} = \varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k + E[\widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}]. \tag{E.2}$$

Together with (E.1), (E.2) *and (24), we obtain the end-to-end distortion for the case where the packet is lost as below*

$$D_{\mathbf{u},ETE}^k\{r,m\} = (f_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}}^k + \varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k + E[\widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}])^2 + \sigma^2_{\widetilde{\zeta}^{k-1}_{\mathbf{u}+\check{\mathbf{mv}}_{\mathbf{u}}^k}}. \tag{E.3}$$

By definition, we have $\varepsilon_{\mathbf{u}}^k = \hat{e}_{\mathbf{u}}^k - \check{e}_{\mathbf{u}}^k$ *and* $\xi_{\mathbf{u}}^k = \hat{f}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-j} - \hat{f}_{\mathbf{u}+\check{\mathbf{m}}\check{\mathbf{v}}_{\mathbf{u}}^k}^{k-1}$*. So, we obtain* $\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k = \hat{f}_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}+\check{\mathbf{m}}\check{\mathbf{v}}_{\mathbf{u}}^k}^{k-1} - \check{e}_{\mathbf{u}}^k$*, and*

$$D_{\mathbf{u},ETE}^k\{r,m\} = (f_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}+\check{\mathbf{m}}\check{\mathbf{v}}_{\mathbf{u}}^k}^{k-1} - \check{e}_{\mathbf{u}}^k + E[\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{m}}\check{\mathbf{v}}_{\mathbf{u}}^k}^{k-1}])^2 + \sigma^2_{\widetilde{\zeta}_{\mathbf{u}+\check{\mathbf{m}}\check{\mathbf{v}}_{\mathbf{u}}^k}^{k-1}}. \tag{E.4}$$

Note that if the error concealment scheme is to copy the reconstructed pixel value from the previous frame, we have $\varepsilon_{\mathbf{u}}^k + \xi_{\mathbf{u}}^k = \hat{f}_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}}^{k-1}$*.*

Note that the error concealment method is the same for intra mode and inter mode since there is no mode information for decoder if the packet is received in error; hence $\check{\mathbf{m}}\check{\mathbf{v}}_{\mathbf{u}}^k$ *and* $\check{e}_{\mathbf{u}}^k$ *in (E.4) are the same for both intra mode and inter mode. On the other hand, the value of* $f_{\mathbf{u}}^k$ *is known before the mode decision and all other variables in (E.4) come from the previous frame. Therefore, the resulting end-to-end distortion in this case, i.e.,* $D_{\mathbf{u},ETE}^k\{r,m\}$ *will also be the same for both intra mode and inter mode.*

Second, if the packet is correctly received, from (20) we obtain $D_{\mathbf{u}}^k\{\bar{r},\bar{m}\} = D_{\mathbf{u}}^k(p)$ *and from (21) we obtain* $E[\widetilde{\zeta}_{\mathbf{u}}^k]\{r,m\} = E[\widetilde{\zeta}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r},\bar{m}\}]$*. From (24), we obtain the end-to-end distortion as*

$$D_{\mathbf{u},ETE}^k\{\bar{r},\bar{m}\} = (f_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}}^k)^2 + D_{\mathbf{u}}^k(p) + 2(f_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}}^k)\cdot E[\widetilde{\zeta}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r},\bar{m}\}]. \tag{E.5}$$

Since both $P_{\mathbf{u}}^k$ *and* $D_{\mathbf{u},ETE}^k\{r,m\}$ *are the same for all modes, we can denote* $P_{\mathbf{u}}^k \cdot D_{\mathbf{u},ETE}^k\{r,m\}$ *by* $C_{\mathbf{u}}^k$*, which is independent of all modes. Let* $D_{\mathbf{u},\overline{ETE}}^k = (1-P_{\mathbf{u}}^k)\cdot D_{\mathbf{u},ETE}^k\{\bar{r},\bar{m}\}$*; then we have*

$$D_{\mathbf{u},ETE}^k(\omega_m) = D_{\mathbf{u},\overline{ETE}}^k(\omega_m) + C_{\mathbf{u}}^k, \tag{E.6}$$

where

$$\begin{aligned} D_{\mathbf{u},\overline{ETE}}^k(\omega_m) &= (1-P_{\mathbf{u}}^k)\cdot D_{\mathbf{u},ETE}^k\{\bar{r},\bar{m}\} && \text{(E.7)}\\ &= (1-P_{\mathbf{u}}^k)\cdot\{(f_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}}^k)^2 + D_{\mathbf{u}}^k(p) + 2(f_{\mathbf{u}}^k - \hat{f}_{\mathbf{u}}^k)\cdot E[\widetilde{\zeta}_{\mathbf{u}+\mathbf{mv}_{\mathbf{u}}^k}^{k-j} + \widetilde{\Delta}_{\mathbf{u}}^k\{\bar{r},\bar{m}\}]\} && \text{(E.8)} \end{aligned}$$

Appendix F. Proof of Proposition 2

Proof 3. *From* (28)*, we have*

$$\begin{aligned} \arg\min_{\omega_m}\{\hat{D}_{ETE}^k(\omega_m) + \lambda\cdot R(\omega_m)\} &= \arg\min_{\omega_m}\{\sum_{\mathbf{u}\in\mathcal{V}_i^k}[D_{\mathbf{u},\overline{ETE}}^k(\omega_m) + C_{\mathbf{u}}^k] + \lambda\cdot R(\omega_m)\}\\ &= \arg\min_{\omega_m}\{\sum_{\mathbf{u}\in\mathcal{V}_i^k} D_{\mathbf{u},\overline{ETE}}^k(\omega_m) + \sum_{\mathbf{u}\in\mathcal{V}_i^k} C_{\mathbf{u}}^k + \lambda\cdot R(\omega_m)\}\\ &= \arg\min_{\omega_m}\{\hat{D}_{\overline{ETE}}^k(\omega_m) + \lambda\cdot R(\omega_m)\}\\ &= \hat{\omega}_m. \end{aligned} \tag{F.1}$$

That is, Algorithm A and Algorithm 2 produce the same solution.

Appendix G. Details of Complexity Comparison

Appendix G.1. RMPC-MS Algorithm

Let us first consider the complexity of RMPC-MS algorithm, i.e., Algorithm 2, for inter modes. In Algorithm 2, the worst case is $\hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}] < \hat{f}^k_{\mathbf{u}} - 255$. Under this case, there are one ADD and one square, i.e., MUL. The other two cases require only two copy operations, and so are neglected. Note that $\hat{f}^k_{\mathbf{u}} - \hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}] \leq 255$ with high probability, that is, $\hat{E}[\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}}] < \hat{f}^k_{\mathbf{u}} - 255$ is relatively rare. Therefore, in most cases, there are only two copy operations in the loop. Calculating the second moment of $\widetilde{\zeta}^k_{\mathbf{u}}$ needs 4 ADDs and 4 MULs as in (20). Similarly, calculating the first moment of $\widetilde{\zeta}^k_{\mathbf{u}}$ needs 2 ADDs and 2 MULs as in (21). Finally, calculating the end-to-end distortion needs 3 ADDs and 2 MULs as in (24). Hence, the worst case of calculating the end-to-end distortion for each pixel is 10 ADDs and 9 MULs. Note that in most cases, the complexity is 9 ADDs and 8 MULs for inter mode as shown in Table 1.

Note that since $P^k_{\mathbf{u}}$ is the same for all pixels in one MB, we do not need to calculate $1 - P^k_{\mathbf{u}}$ for each pixel. Multiplying by 2 can be achieved by a shift operation; so it is not counted as one MUL.

For Intra modes, we know that $\widetilde{\zeta}^{k-j}_{\mathbf{u}+\mathbf{mv}^k_{\mathbf{u}}} + \widetilde{\Delta}^k_{\mathbf{u}}\{\bar{r}, \bar{m}\} = 0$ from Ref. [1]. Therefore, the complexity of intra mode is reduced to 3 ADDs and 3 MULs in (20), 1 ADDs and 1 MULs in (22). As a result, the end-to-end distortion for each pixel is 7 ADDs and 6 MULs for each intra mode.

In H.264, there are 7 inter modes and 13 intra modes; therefore there are a total of 154 ADDs and 134 MULs for each pixel in most cases. In the worst case, there are a total of 161 additions and 141 MULs for each pixel, where the additional computation comes from the consideration of clipping effect.

Memory Requirement Analysis: To estimate the end-to-end distortion by Algorithm 2, the first moment and the second moment of the reconstructed error of the best mode should be stored after the mode decision. Therefore, 2 units of memory are required to store those two moments for each pixel. Note that the first moment takes values in $\{-255, -254, \cdots, 255\}$, i.e., 8 bits plus 1 sign bit per pixel, and the second moment takes values in $\{0, 1, \cdots, 255^2\}$, i.e., 16 bits per pixel[8].

Appendix G.2. ROPE Algorithm

In ROPE algorithm, the moment estimation formulae for inter prediction and intra prediction are different. For inter mode, calculating the first moment needs 2 ADDs and 2 MULs; calculating the second moment needs 3 ADDs and 4 MULs; calculating the end-to-end distortion needs 2 ADDs and 2 MULs. For intra mode, calculating the first moment needs 1 ADD and 2 MULs; calculating the second moment needs 1 ADD and 3 MULs.

[8]The definitions of these variables have been released in JM17.0 [28]. However, some related algorithms are still under the patent filing process.

Hence, an inter mode needs 7 ADDs and 8 MULs; an intra mode needs 4 ADDs and 7 MULs. For H.264, the total complexity for each pixel is 101 ADDs and 147 MULs.

Note that when we implement ROPE in JM16.0 [15], we find that ROPE algorithm causes out-of-range values for both the first moment and the second moment due to the neglect of clipping noise. Experimental results show that ROPE may produce a negative value as the estimate for distortion, which violates the requirement that (true) distortion must be non-negative. Hence, in a practical system, the estimate obtained by ROPE algorithm needs to be clipped into a legitimate value; this will incur a higher complexity.

Memory Requirement Analysis: To estimate the end-to-end distortion by ROPE algorithm, the first moment and the second moment of the reconstructed pixel value of the best mode should be stored after the mode decision. Therefore, 2 units of memory are required to store the two moments for each pixel. The first moment takes values in $\{0, 1, \cdots, 255\}$, i.e., 8 bits per pixel; the second moment takes values in $\{0, 1, \cdots, 255^2\}$, i.e., 16 bits per pixel. Note that in the original ROPE algorithm [8], the values of the two moments are not bounded since the propagated errors are never clipped.

Appendix G.3. LLN Algorithm

In JM16.0 [15], LLN algorithm uses the same decomposition method as Theorem 1 for mode decision [12]. In such a case, for inter modes, reconstructing the pixel value in one simulated decoder needs 1 ADD; calculating the end-to-end distortion needs 1 ADD and one MUL. For intra modes, there is no additional reconstruction for all simulated decoders since the newly induced errors are not considered; therefore, calculating the end-to-end distortion needs 1 ADD and 1 MUL. Suppose the number of simulated decoders at the encoder is N_d, the complexity for LLN algorithm is $27N_d$ ADDs and $20N_d$ MULs. The default number of simulated decoders in JM16.0 is 30, which means the complexity for LLN algorithm is 810 ADDs and 600 MULs. Thirty simulated decoders is suggested in Ref. [29]. In our experiment, we find that if the number of simulated decoders is less than 30, the estimated distortion exhibits high degree of randomness (i.e., having a large variance); however, if the number of simulated decoders is larger than 50, the estimated distortion is quite stable (i.e., having a small variance).

Note that the error concealment operations in LLN algorithm are required but not counted in the complexity since it is done after the mode decision. However, even without considering the extra error concealment operations, the complexity of LLN algorithm is still much higher than RMPC-MS and ROPE. Increasing the number of simulated decoders at the encoder can improve estimation accuracy but at the cost of linear increase of computational complexity.

Memory Requirement Analysis: To estimate the end-to-end distortion by LLN algorithm, for each simulated decoder, each reconstructed pixel value of the best mode should be stored after the mode decision. Therefore, the encoder needs N_d units of memory to store the reconstructed pixel value. A reconstructed pixel takes values in $\{0, 1, \cdots, 255\}$, i.e., $8N_d$ bits per pixel.

References

[1] Z. Chen, D. Wu, Prediction of Transmission Distortion for Wireless Video Communication: Part I: Analysis, http://www.wu.ece.ufl.edu/mypapers/journal-1.pdf (2010).

[2] K. Stuhlmuller, N. Farber, M. Link, B. Girod, Analysis of video transmission over lossy channels, IEEE Journal on Selected Areas in Communications 18 (2000) 1012–1032.

[3] J. U. Dani, Z. He, H. Xiong, Transmission distortion modeling for wireless video communication, in: Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'05), 2005.

[4] Z. He, J. Cai, C. W. Chen, Joint source channel rate-distortion analysis for adaptive mode selection and rate control in wireless video coding, IEEE Transactions on Circuits and System for Video Technology, special issue on wireless video 12 (2002) 511–523.

[5] Y. Wang, Z. Wu, J. M. Boyce, Modeling of transmission-loss-induced distortion in decoded video, IEEE Transactions on Circuits and Systems for Video Technology 16 (6) (2006) 716–732.

[6] Y. J. Liang, J. G. Apostolopoulos, B. Girod, Analysis of packet loss for compressed video: Effect of burst losses and correlation between error frames, IEEE Transactions on Circuits and Systems for Video Technology 18 (7) (2008) 861–874.

[7] H.264/AVC reference software JM14.0 (May. 2008).
URL `http://iphome.hhi.de/suehring/tml/download`

[8] R. Zhang, S. L. Regunathan, K. Rose, Video coding with optimal inter/intra-mode switching for packet loss resilience, IEEE Journal on Selected Areas in Communications 18 (6) (2000) 966–976.

[9] T. Stockhammer, M. Hannuksela, T. Wiegand, H. 264/AVC in wireless environments, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 657–673.

[10] Y. Zhang, W. Gao, Y. Lu, Q. Huang, D. Zhao, Joint source-channel rate-distortion optimization for h.264 video coding over error-prone networks, IEEE Transactions on Multimedia 9 (3) (2007) 445–454.

[11] T. M. Cover, J. A. Thomas, Elements of Information Theory, Wiley-Interscience, 1991.

[12] T. Stockhammer, D. Kontopodis, T. Wiegand, Rate-distortion optimization for JVT/H. 26L video coding in packet loss environment, in: Proc. International Packet Video Workshop, Citeseer, 2002.

[13] YUV video sequences.
URL `http://trace.eas.asu.edu/yuv/index.html`

[14] ITU-T Series H: Audiovidual and Multimedia Systems, Advanced video coding for generic audiovisual services (Nov. 2007).

[15] H.264/AVC reference software JM16.0 (Jul. 2009).
URL `http://iphome.hhi.de/suehring/tml/download`

[16] G. Bjontegaard, Calculation of average PSNR differences between RD-curves, 13th VCEG-M33 Meeting, Austin, USA.

[17] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the h.264/AVC video coding standard, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 560–576.

[18] H. Yang, K. Rose, Advances in recursive per-pixel end-to-end distortion estimation for robust video coding in H. 264/AVC, IEEE Transactions on Circuits and Systems for Video Technology 17 (7) (2007) 845.

[19] J. Ribas-Corbera, S. Lei, Rate control in DCT video coding for low-delay communications, IEEE Transactions on Circuits and Systems for Video Technology 9 (1) (1999) 172–185.

[20] S. Ma, W. Gao, Y. Lu, Rate-distortion analysis for H. 264/AVC video coding and its application to rate control, IEEE Transactions on Circuits and Systems for Video Technology 15 (12) (2005) 1533.

[21] Z. Li, W. Gao, F. Pan, S. Ma, K. Lim, G. Feng, X. Lin, S. Rahardja, H. Lu, Y. Lu, Adaptive rate control for H. 264, Journal of Visual Communication and Image Representation 17 (2) (2006) 376–406.

[22] B. Girod, The efficiency of motion-compensating prediction for hybrid coding of video sequences, IEEE Journal on Selected Areas in Communications 5 (7) (1987) 1140–1154.

[23] G. Casella, R. L. Berger, Statistical Inference, 2nd Edition, Duxbury Press, 2001.

[24] A. Goldsmith, Wireless Communications, Cambridge University Press, 2005.

[25] D. Agrafiotis, D. R. Bull, C. N. Canagarajah, Enhanced error concealment with mode selection, IEEE Transactions on Circuits and Systems for Video Technology 16 (8) (2006) 960–973.

[26] J. Wu, X. Liu, K.-Y. Yoo, A temporal error concealment method for h.264/avc using motion vector recovery, IEEE Transactions on Consumer Electronics 54 (4) (2008) 1880–1885.

[27] Y. Wang, Q.-F. Zhu, Error control and concealment for video communication: a review, in: Proceedings of IEEE, 1998.

[28] H.264/AVC reference software JM17.0 (Feb. 2010).
URL `http://iphome.hhi.de/suehring/tml/download`

[29] T. Stockhammer, T. Wiegand, S. Wenger, Optimized transmission of h.26l/jvt coded video over packet-lossy networks, in: IEEE ICIP, 2002.