An Implementation of PRISM Using QccPack

James E. Fowler

Technical Report MSSU-COE-ERC-05-01

January 2005

GeoResources Institute—Mississippi State ERC
Mississippi State University

Abstract

The current state of an implementation of the PRISM video coder based on the open-source QccPack library is described. Experimental results are presented that compare the PRISM implementation to the simple intraframe coding of video.

1. Introduction

This document describes the current status of an implementation of the PRISM video-coding system as initially proposed in [1] and described in somewhat greater detail in [2]. The implementation was developed so as to follow the description of PRISM in [2] as closely as possible, although they are a few components that have purposely been implemented in a slightly different manner (see Sec. 4 below). The implementation relies heavily on the video-processing, error-correcting-code, and entropy-coding components of the QccPack library [3]; most notably, version 0.48 of QccPack is relied upon to provide Viterbi decoding, trellis-code syndrome calculation, scalar quantization, DCT, Huffman coding, and image-sequence input/output.

2. Implementation

The PRISM system is implemented in three distinct modules: training, encoding, and decoding. Throughout, we use 8×8 blocks, the Ungerboeck 128-state rate-1/2 trellis code from [4], and a 16-bit CRC.

1. Training—The PRISM training component estimates the correlation noise which will later guide the quantization of inter blocks. In addition, the training procedure determines the optimal linear estimator for reconstruction of the inter blocks.

   (a) Correlation-Noise Estimation—For each inter block in the current frame, we determine its class (see 2.1 below). We then calculate the square of the difference between coefficient i of the block in the current frame and corresponding coefficient of the matching block in the reference frame, where the matching block is found via full-search motion estimation (MAE metric) at half-pixel accuracy, and the coefficients are indexed in zig-zag scan order. The average of the squared differences (over all inter blocks of all frames) are tabulated as $\sigma[c, i]$ for each class c and each coefficient i, and the resulting $\alpha[c, i] = \sqrt{\sigma[c, i]}$ values are stored in a training-data file. These α values drive the quantization of the inter-block coefficients according to the "inverse-waterfilling" algorithm as applied to the innovations coefficients as dictated by [5].

   (b) Optimal Linear Estimation—For each coefficient i and for each class c, we determine the optimal linear estimator for that coefficient by estimating the two blocks produced by the decoder (see 3.2.1 below): one block from the full-search motion estimation in 1.1 above and the other block by mimicking the quantization produced by the Viterbi decoding and the refinement processes in 3.1.1.1 and 3.2.2, respectively. The optimal linear estimator is determined by calculating autocorrelations of the various quantities and solving the resulting Yule-Walker equations. We end up with weights $a_0[c, i]$ and $a_1[c, i]$ for class c and coefficient i to be used in optimal estimation in 3.2.3 below.

2. Encoder—Input parameter: q (desired quantization stepsize). A block diagram of the encoder architecture is depicted in Fig. 1.

   (a) Classification—Each 8×8 block is classified into one of 16 classes. The squared error between the current block and the co-located block in the previous frame is calculated. If the squared error is less than `THRESHOLD_SKIP`, the block is labeled as "skip" and is not coded. If the squared error is greater than `THRESHOLD_INTRA`, the block is labeled as "intra" and is intracoded (see 2.3 below). Otherwise, the block is labeled as "inter" and is syndrome-coded. The inter blocks are classified into one of 16 classes using a 16-level uniform scalar quantizer applied to the squared error between the current block and the co-located block in the previous frame. The block type (skip, intra, inter) is Huffman coded, and, for inter blocks, the class is coded with a 4-bit fixed-length binary code. Currently, `THRESHOLD_SKIP` and `THRESHOLD_INTRA` are hardcoded as 18.33 and 8186.0, respectively.

   (b) Decorrelation Transform—Intra and inter blocks are subjected to an 8×8 DCT.

   (c) Intra Blocks—All blocks of the first frame of the sequence are intra coded, as are blocks from subsequent frames identified by the classifier as intra blocks

      i. Quantization—DCT coefficients of the 8×8 intra block are quantized using uniform scalar quantizers. The stepsizes of these quantizers are $q'[i] = q \cdot Q[i]$, where $Q[i]$ is the JPEG Annex K quantization matrix, zig-zag scanned (this modulation of the Q matrix by q mimics the "quality factor" quality-control mechanism common to JPEG implementations).

      ii. Huffman Coding—The DC coefficient of the 8×8 intra block is Huffman coded using the JPEG Annex K DC Huffman table, followed by zig-zag runlength Huffman coding of the AC coefficients using the JPEG Annex K AC Huffman table.

   (d) Inter Blocks

      i. Base Quantization—The DC coefficient and the first 14 AC coefficients (in zig-zag scan order) in the 8×8 DCT inter block are quantized with a uniform scalar quantizer of stepsize $q''[i] = s \cdot \alpha[c, i]$, where c is the class of the block (from the classifier), i is the index of the coefficient within the block, and s is a scaling constant (currently, s is hardcoded to the value determined empirically to yield the best results for the given sequence).

      ii. Syndrome—The codewords for the 15 Wyner-Ziv (WZ) coefficients of the inter block are calculated by taking the quantization indices mod 4. The 15-bit syndrome is calculated by convolving the mod-4 codewords with the parity-check matrix of the trellis code.

      iii. CRC—A 16-bit CRC is calculated on the 15 quantization indices of the inter block.

      iv. Refinement—The desired stepsize for coefficient i is $q'[i] = q \cdot Q[i]$, where $Q[i]$ is the JPEG Annex K quantization matrix in zig-zag scan order. We divide the base quantization interval of width $q''[i]$ into $n[i] = \frac{q''[i]}{q'[i]}$ refinement intervals and send an $l[i]$-bit fixed-length binary code as refinement information, where $l[i] = \mathrm{rint}\big(\log_2(n[i])\big)$. This is done for each of the 15 WZ coefficients.

      v. Intra Coding—The remaining 49 DCT coefficients in the inter block are intra coded as described above in 2.3.

3. Decoder—The block diagram of the PRISM decoder architecture is illustrated in Fig. 2.

   (a) Intra Blocks—The coefficients of an intra block are decoded (runlength Huffman decoding and inverse quantization).

   (b) Inter Blocks

      i. Motion Search—For each inter block, a half-pixel motion search searches for a matching block. This search starts at the $(0, 0)$ motion vector and "spirals" outward with ever-increasing radius until a ± 15-pixel window is searched.

         A. Syndrome Decoding—Each candidate matching block is fed into a Viterbi decoder using soft decoding. The Viterbi decoder takes as input the 15-bit syndrome sequence as well as the 15 DCT coefficients of the candidate block and outputs the 15 4-mod codewords of the closest codeword sequence with the desired syndrome.

         B. CRC Check—The 15 mod-4 codewords from the Viterbi decoder are used in conjunction with the 15 DCT coefficients from the candidate block to recover the 15 quantization indices for the current block. To do so, for each coefficient, we find the reconstructed value with the proper mod-4 label that is closest to the corresponding coefficient from the candidate block. We then generate the 16-bit CRC for the quantization indices and compare to the CRC transmitted by the encoder. If the CRCs match, the motion search terminates and we go to 3.2.2. If the CRCs do not match we move on to the next candidate block and repeat 3.2.1.1.

      ii. Refinement—After the matching block is found, we refine the value of the 15 WZ coefficients using the $l[i]$-bit fixed-length binary code as refinement information.

      iii. Estimation and Reconstruction—We now have two estimates of the 15 WZ coefficients of the current block: $y_0[i]$ are the coefficients from the matching block in the reference frame, $y_1[i]$ are the coefficients reconstructed by Viterbi decoding and refinement. We form the final estimate of the current block as $x[i] = a_0[c, i]y_0[i] + a_1[c, i]y_1[i]$, where a_0 and a_1 are the optimal linear estimator values from the training procedure (see 1.2 above). This procedure is applied for all 15 WZ coefficients of the current block.

      iv. Intra Decoding—The remaining 49 coefficients are intra decoded (runlength Huffman decoding and inverse quantization).

   (c) Inverse Transform—Inter and intra blocks are subject to an inverse 8×8 DCT.

   (d) Error Concealment—In the case that the motion search fails to find a match for an inter block, we use bilinear interpolation from the pixels on the boundaries of the neighboring blocks to reconstruct an estimate of the missing block following the procedure described in [6].

3. Rate-Distortion Results

Fig. 3 shows the rate-distortion performance of the PRISM implementation as compared to intraframe coding for the first 16 frames of the "Football" sequence. Distortion is measured as the PSNR averaged over all 16 frames of the sequence. For the PRISM implementation, the first frame is entirely intracoded, while the subsequent 15 frames are coded using a combination of skip, intra, and inter blocks (approximate distribution: 33% skip, 56% intra, 11% inter). The quantization scaling factor is set to $s = 7$. Typically, the decoder fails to find matches for less than 0.5% of the inter blocks. For the intraframe results, the PRISM implementation is used with all blocks in all frames forced to be coded as intra blocks.

Fig. 4 shows the rate-distortion performance of the PRISM implementation for the first 16 frames of the "Susie" sequence. For this sequence, $s = 10$, and the approximate distribution of blocks is 66% skip, 33% intra and 1% inter. Like above, less than 0.5% of the inter blocks typically are unmatched by the decoder.

4. Known Implementation Differences

While the current implementation attempts to follow the description of [2] as closely as possible, there are several aspects in which the implementation is known to deviate from the description.

1. For inter blocks, we apply the CRC to the quantization indices of the WZ coefficients rather than to the mod-4 codewords since this resulted in more reliable matches and slightly better performance.

2. In our PSNR calculations, we calculate a full-frame PSNR. That is, PSNR values include blocks for which the motion search failed and which have had to be reconstructed via the error-concealment process. The results in the PRISM paper apparently excluded such failed-match blocks from the PSNR figures despite the fact that, presumably, some form of error concealment must have been implemented so as to produce an intact reference frame for the decoding of the next frame.

3. To compare to "intraframe coding," the PRISM paper apparently uses H.263+ in "intraframe mode;" i.e., with all I frames. Our "intraframe coding" is our PRISM coder with all blocks forced to be coded as intra (i.e., no skip or inter blocks). This is essentially motion JPEG. We do not know if there is a substantial performance difference between H.263+ with I frames and motion JPEG.

Acknowledgment

This work was funded in part by by the French Centre National de la Recherche Scientique and by the US National Science Foundation under Grant No. CCR-0310864. This work was conducted from May to December, 2004, while the author was a visiting professor in the Département TSI at École Nationale Supérieure des Télécommunications (ENST), Paris, France, while on sabbatical leave from Mississippi State University.

References

[1] R. Puri and K. Ramchandran, “PRISM: A new robust video coding architecture based on distributed compression principles,” in *Proceedings of the Allerton Conference on Communication, Control, and Computing*, Urbana-Champaign, IL, October 2002.

[2] R. Puri and K. Ramchandran, “PRISM: A video coding architecture based on distributed compression principles,” Tech. Rep. No. UCB/ERL M03/6, ERL, UC Berkeley, March 2003.

[3] J. E. Fowler, “QccPack: An open-source software library for quantization, compression, and coding,” in *Applications of Digital Image Processing XXIII*, A. G. Tescher, Ed., San Diego, CA, August 2000, Proc. SPIE 4115, pp. 294–301.

[4] G. Ungerboeck, “Channel coding with multilevel/phase signals,” *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, January 1982.

[5] S. S. Pradhan, “On rate-distortion function of Gaussian sources with memory in the presence of side information at the decoder,” Tech. Rep., Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, December 1998.

[6] P. Salama, N. B. Shroff, E. J. Coyle, and E. J. Delp, “Error concealment techniques for encoded video streams,” in *Proceedings of the International Conference on Image Processing*, Washington, DC, October 1995, vol. 1, pp. 9–12.

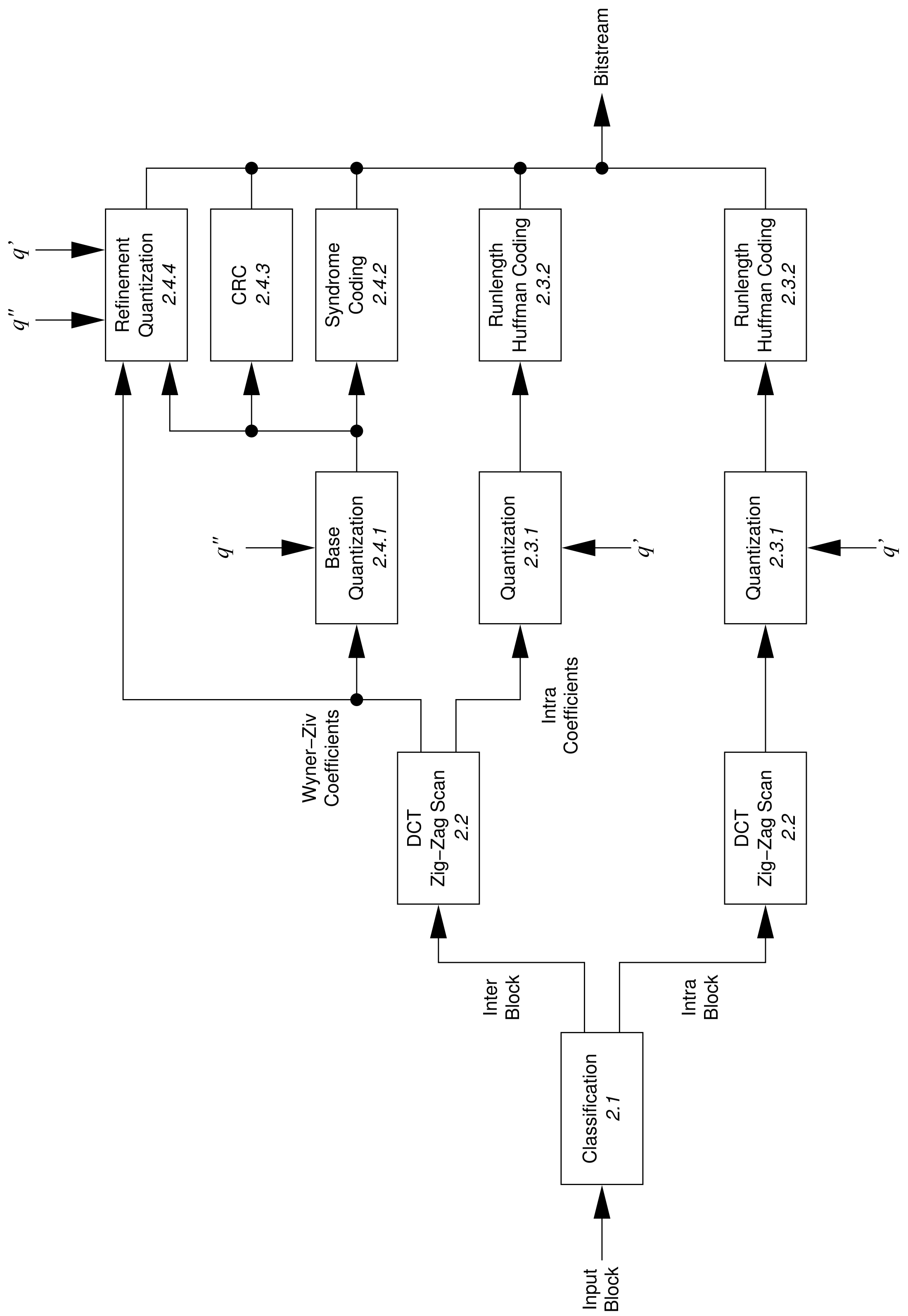

Figure 1: The encoder of the PRISM implementation (adapted from Fig. 8 of [2]). Numbers within the blocks indicate components of the description here.

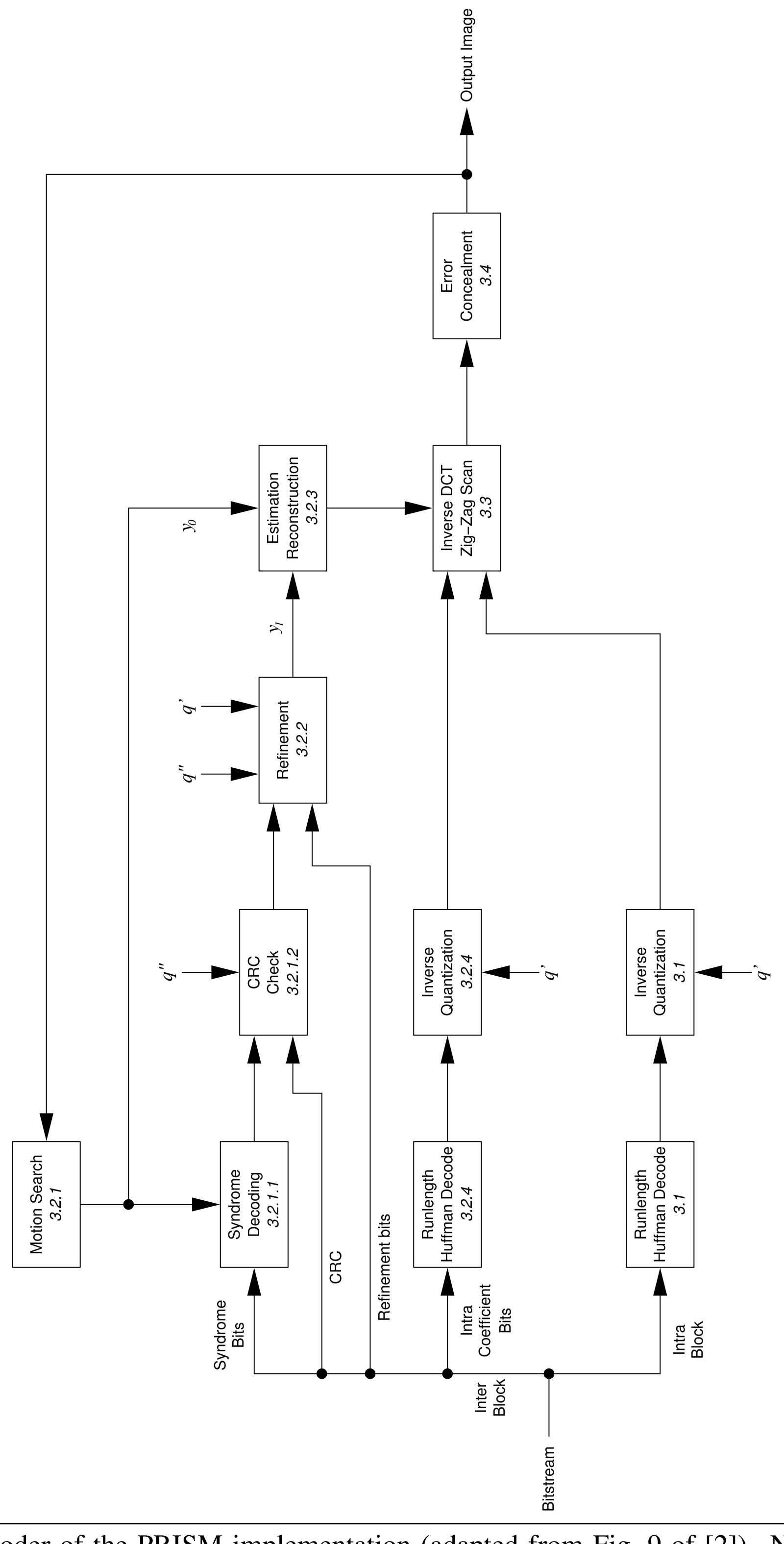

Figure 2: The decoder of the PRISM implementation (adapted from Fig. 9 of [2]). Numbers within the blocks indicate components of the description here.

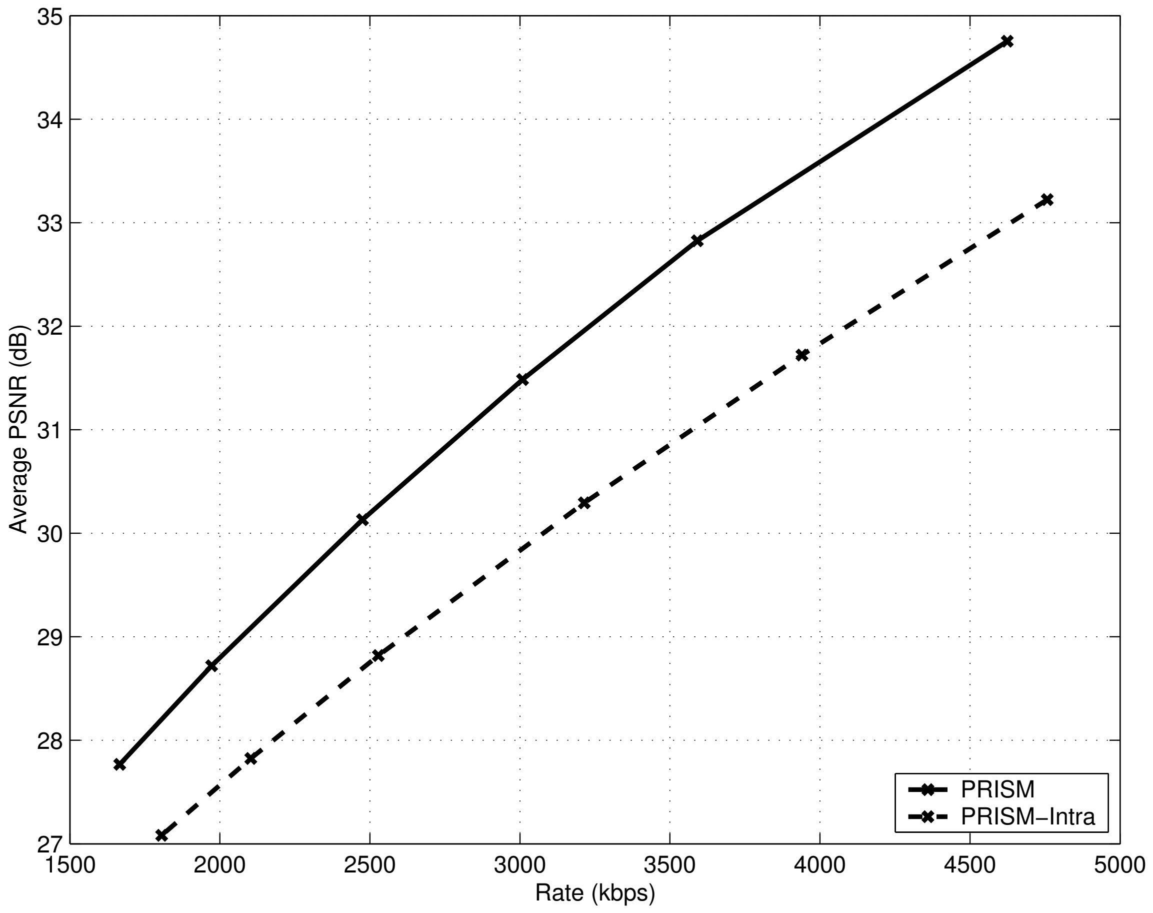

Figure 3: Rate-distortion performance for the first 16 frames of “Football.”

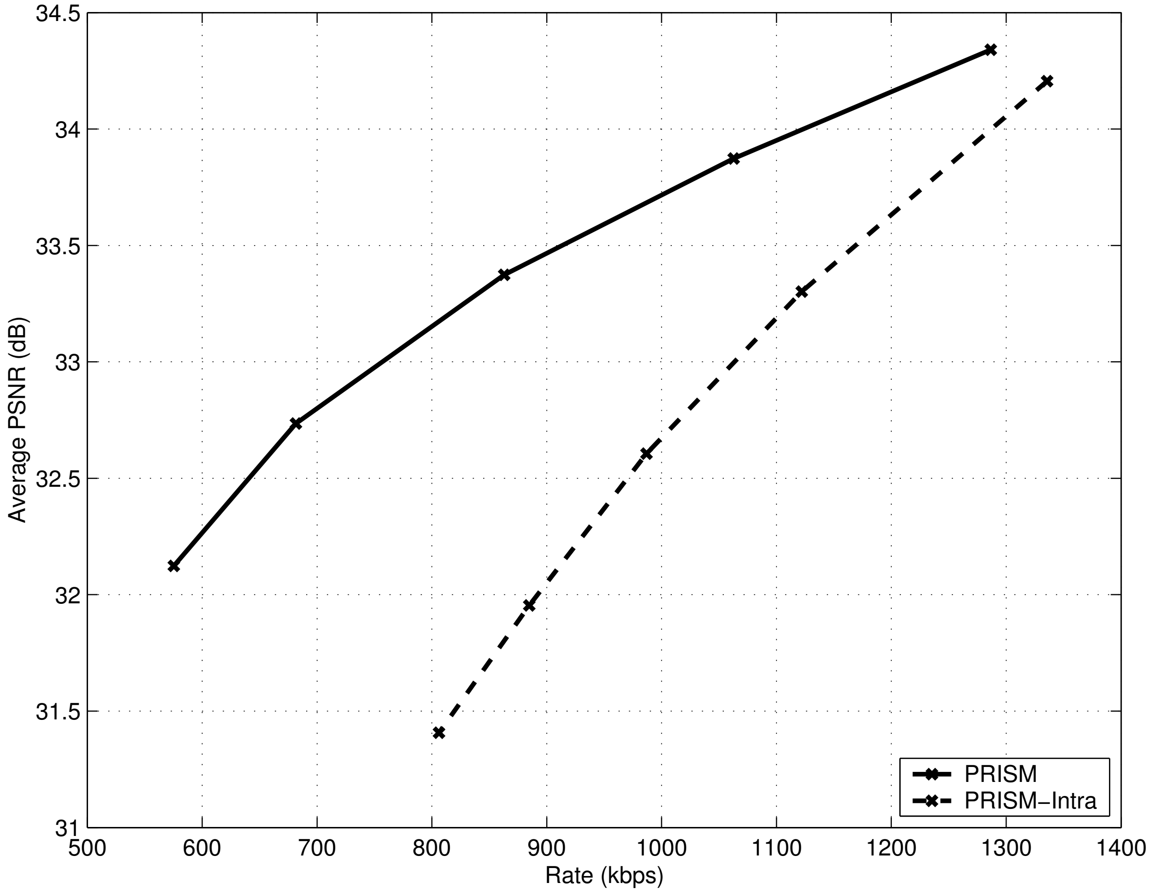

Figure 4: Rate-distortion performance for the first 16 frames of “Susie.”