Predictive Patch Matching for Inter Frame Coding

Tianmi Chen[1*], Xiaoyan Sun[2], and Feng Wu[2]
[1]Xidian University, Xi'an, China;
[2]Internet Media Group, Microsoft Research Asia, Beijing, China

ABSTRACT

Inter prediction is an important component for video coding which exploits the temporal correlation between frames and significantly reduces the redundancy in video sequences. In this paper, we propose a predictive patch matching for inter prediction based on template matching prediction. Besides the surrounding reconstructed pixels which form the template in template matching prediction, our proposed patch matching is able to utilize the predicted pixels generated by the traditional motion prediction. A linear combination of the reconstructed template and the predicted pixels permits to synthesize a prediction while maintaining the local variations of the target block. Furthermore, a mode selection mechanism is introduced to adaptively select the predictive patch matching at sub-block level. Experimental results demonstrate the effectiveness of our proposed predictive patch matching. Constant coding gain can be achieved by our scheme at both low and high bit rates.

Keywords: Video coding, inter prediction, template matching, patch matching

1. INTRODUCTION

The huge number of contents as well as the wide range of topics has turned digital video consuming and sharing into one of the most important parts of internet service. Video compression, as an essential component, becomes more and more important in facilitating the video consuming especially for the videos at high resolution and quality levels. Generally speaking, a traditional video coding scheme consists of five modules - intra prediction, inter prediction, transform, quantization and entropy coding. Among these modules, inter prediction exploits the temporal correlation between frames and significantly reduces the redundancy in video sequences.

Many researchers have put efforts in developing advanced inter prediction methods [1]. A well-known and widely adopted approach is block-based motion prediction (BMP). In this approach, a video frame is divided into non-overlapped blocks. For each target block, a predicted block is selected from its adjacent frames by a block matching mechanism. The displacement between the predicted block and the target block, also known as the motion vector, is transmitted to the decoder. To improve the performance of block-based prediction, adaptive block partitioning has been introduced in a rate-distortion optimal fashion. Also, the degree of motion prediction has been extended from integer pixel position to fractional one. Consequently, in order to estimate and compensate the pixel values at the fractional displacements, adaptive interpolation filters are presented for reducing aliasing in the predicted signals. Benefit from these approaches the state-of-the-art H.264/AVC coding standard achieves significant coding gain in inter frame compression [2].

Besides the block-based solutions, some vision techniques, such as texture synthesis [3], are applicable to exploit the visual redundancy in images and videos. Promising results have been reported by employing texture synthesis in compression even in a straight-forward fashion [4]. Improved perceptual quality is achieved by using spatio-temporal texture synthesis and edge-based inpainting for video coding [5][6]. Recently, the template matching (TM), which is widely used in texture synthesis, has been integrated with the block-based prediction for inter frame coding [7]. By finding the predicted pixels using their surrounding reconstructed pixels, the template matching prediction (TMP) generates a prediction without transmitting the motion vector. A smoother prediction is produced by averaging the

[*] This work has been done during T. Chen's internship in Microsoft Research Asia.

Visual Communications and Image Processing 2010, edited by Pascal Frossard, Houqiang Li, Feng Wu, Bernd Girod, Shipeng Li, Guo Wei, Proc. of SPIE Vol. 7744, 774412 · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.863461

multiple candidate predictions in TMP [8]. Also, the displacement information estimated in TMP can be used as a motion vector predictor to save the motion bits in inter frame coding [9].

In this paper, we propose a predictive patch matching (PPM) for inter prediction. It is inspired by the TMP. However, different from the TMP, the proposed PPM utilizes not only the correlations between templates but also the motion compensated pixels. In fact, it can be regarded as a combination of TMP and traditional motion prediction. For each target block, the traditional motion prediction is performed first. The compensated block along with its surrounding template is used in searching the candidate predictive patches based on the motion information. Then the PPM predicted block is generated by averaging the block regions in the candidate patches. The PPM prediction is selectively utilized in the presented coding scheme at sub-block level in a rate-distortion optimal way. Our primary experimental results show that by integrating our proposed PPM, the improved H.264/AVC coding scheme achieves 0.2dB gain in terms of PSNR.

2. TEMPLATE MATCHING PREDICTION

The TMP exploits the self-similarity between different frames within a video sequence. Let $F = \{f_h\}$ represents an input video, where the subscript h denotes the time stamp of each frame. Given the current frame f_h and its reference frame $\hat{f}_{h-1}$, the basic idea of TMP is illustrated in Fig. 1.

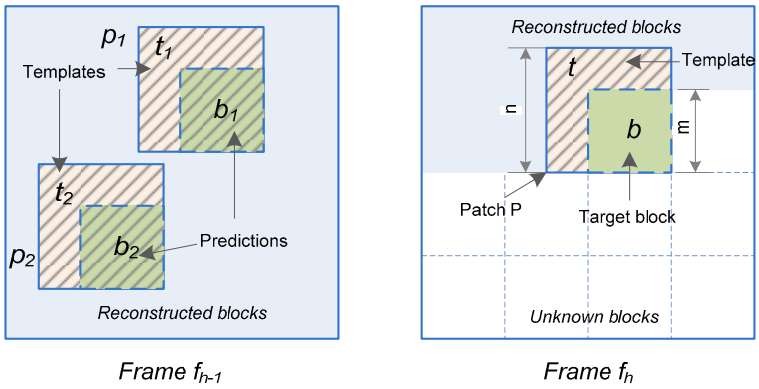


Fig. 1. Illustration of TMP. The blue and white regions denote the reconstructed and unknown regions, respectively. In this figure, the current frame is $\mathbf{f_h}$. The target block is $\mathbf{b}$ and its template is denoted by $\mathbf{t}$. The candidate patches are denoted by $\mathbf{p_1}$and $\mathbf{p_2}$ in the reference frame $\hat{\mathbf{f}}_{\mathbf{h-1}}$.

The basic process unit in TMP is an n × n patch p. As denoted in Fig. 1, a patch consists of two parts, an unknown target block b (of size m × m) and the surrounding inverse L-shaped template t filled with reconstructed pixels.

$$p = b \cup t \text{ and } b \cap t = \emptyset. \tag{1}$$

For each target block b, a candidate patch p_i is selected by TMP if

$$t_i = \text{argmin}_{p_j \in \hat{f}_{h-1}} \Phi(t_j, t) \text{ , where } p_j = b_j \cup t_j \text{ and } b_j \cap t_j = \emptyset. \tag{2}$$

Here $\Phi(\cdot)$ measures the distance between two patches at template regions. If only one candidate is selected, the prediction of block b is b_i. Otherwise, when the first I-candidate patches with the lowest distance to the template t are selected, let's say $\{p_i\}_I$, the prediction of block b becomes the combination of the blocks $\{b_i\}_I$. For example, the prediction is generated by averaging the multiple predictive blocks in [8].

The advantage of the TMP is clear. It measures the distance by surrounding reconstructed pixels rather than original pixels, thus no motion information is needed. Same prediction is achievable by employing the TMP process at the decoder side. Also, it is able to estimate the prediction at sub-block or even pixel level. On the other hand, the efficiency of the TMP highly depends on the correlation between the template and the target block. In other words, the TMP is based on the assumption that

$$\Phi(b_j, b) \propto \Phi(t_j, t). \tag{3}$$

However, for natural video sequences, this assumption is invalid. When this correlation is weak, it is hard for the TMP to generate a comparable prediction to the one estimated by the BMP.

3. PREDICTIVE PATCH MATCHING

In this section, we propose the PPM for inter prediction. It combines the traditional BMP with the TMP.

3.1 Predictive patch

Similar to the TMP, the basic unit in our PPM is a $n \times n$ patch p, which consists of a target block b and its surrounding template t. Differently, during the PPM prediction, the target block b is first filled with the predicted pixels generated by the traditional BMP. Then the whole patch p instead of the template t is utilized in searching the candidate patch. supposing the current frame is f_h and its reference frame is $\hat{f}_{h-1}$, the candidate patch p_i is determined as

$$p_i = \operatorname{argmin}_{p_j \in \hat{f}_{h-1}} \phi(p_j, p), \tag{4}$$

where

$$\phi(p_j, p) = \alpha \cdot \Phi(t_j, t) + \beta \cdot \Phi(b_j, \tilde{b}_{bmp}). \tag{5}$$

Here p and p_j are same as those in (1) and (2). α and β in (5) are weighting factors that represent the importance of the template and predicted block in the patch matching. $\tilde{b}_{bmp}$ denotes the predicted block generated by the traditional BMP. If only one candidate patch p_i is selected, then b_i becomes the prediction of target block b where $p_i = \{b_i, t_i\}$.

In this paper, we call the patch filled with a template and a predicted block a predictive patch. It can be observed that using the predictive patches, our PPM can be treated as a linear combination of PPM and TMP. Especially, when $\alpha = 1$ and $\beta = 0$, the PPM performs as same as the TMP; on the contrary, it results in the same prediction as BMP when $\alpha = 0$ and $\beta = 1$. We believe that the weighting factors in (4) should be adaptive to the spatial correlation between the template and the target block inside each patch. However, in our current coding scheme, we set $\alpha = \beta = 0.5$ to test the primary performance of the PPM.

3.2 Motion-based matching

To further improve the performance of PPM, we propose to generate multiple candidates based on motion threading. As indicated in subsection 3.1, for each target block, the BMP is performed and resulting a block-based motion vector. This motion vector guides the route of our PPM, as shown in Fig.2.

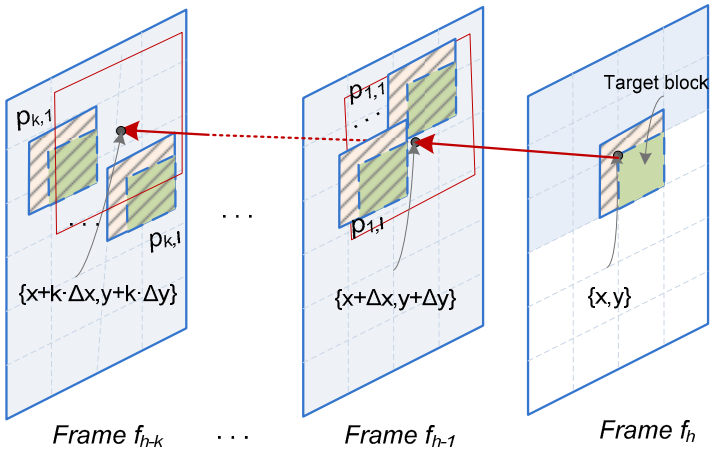


Fig. 2. Motion-based matching in our PPM. The blue and white regions denote the reconstructed and unknown regions, respectively. The current frame is $\mathbf{f_h}$ and the target block is shown by a green square. In the exemplified PPM, **K** reference frames, $\hat{\mathbf{f}}_{\mathbf{h-1}}$ to $\hat{\mathbf{f}}_{\mathbf{h-K}}$, are presented. The red lines with arrows represent the motion threading. The motion based search regions are denoted by the red squares. $\mathbf{p_{k,i}}$ presents the i-th candidate patch in the search range of k-th frame.

In the PPM, the mapping process should take both the temporal motion and the spatial template correlation into consideration. In our solution, the search range of patch matching is determined by the motion vector and the candidates are selected by matching patches inside the range. As shown in Fig. 2, supposing the target block positions at $\{x, y\}$ in f_h, the motion vector of block b at reference frame $\hat{f}_{h-1}$ is $\{\Delta x, \Delta y\}$. Then the center of patch matching in frame $\hat{f}_{h-1}$ is $r_{h-1} = \{x + \Delta x, y + \Delta y\}$. When multiple references are available, e.g. K reference frames, the search center of k-th reference is

$$r_{h-k} = \{x + k \cdot \Delta x,\ \ y + k \cdot \Delta y\},\ \ k = 1,2, \dots, K. \tag{6}$$

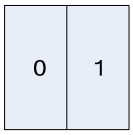
In case the search range S is constant and equally I candidate patches are extracted from each reference frames, totally $K \times I$ candidate patches $\{p_{k,i}\}_{K,I}$ are selected. Along with the motion predicted block $\tilde{b}_{bmp}$, the final prediction of b in our PPM is

$$\tilde{b}_{ppm} = \frac{1}{K \times I + 1}\left(\sum_{k=1}^{K}\sum_{i=1}^{I} b_{k,i} + \tilde{b}_{bmp}\right). \tag{7}$$

3.3 Sub-block compensation

The PPM prediction is selectable at sub-block level. Similar to H.264/AVC, our PPM prediction supports 16×16, 16×8, 8×16 and 8×8 partitions. In case of 8×8 partition, smaller sizes including 8×4, 4×8 and 4×4 are considered. (Detailed segmentation of macroblock for motion compensation can be found in [2].) So our proposed PPM can be readily integrated into the H.264/AVC coding scheme.

Taking an 8×16 predictive mode as an example, a macroblock is partitioned into two 8×16 regions as denoted in Fig.3. If the PPM mode is enabled, one-bit flag is required at macroblock level indicating whether the PPM is used for compensating the macroblock or not. For each 8×16 region, another one-bit flag is introduced to signal whether the region is compensated using PPM or the traditional BMP. In order to reduce these overhead bits, the Content-Adaptive Binary Arithmetic Coding (CABAC) [10] is used in coding the flags.



8×16

Fig. 3. Illustration of 8x16 PPM compensation.

For each macroblock, the predictive mode is determined by minimizing the distortion

$$J = D + \lambda \cdot R, \tag{8}$$

where D measures the distortion between original and predicted block, λ denotes the Lagrange parameter, and R represents the bits for coding the motion vectors, residues, and predictive mode.

4. EXPERIMENTAL RESULTS

To validate the effectiveness of our proposed PPM, we integrate the PPM into the H.264/AVC reference software JM11.0 [11]. Two test sequences, Foreman (176x144) at 30fps and BQSquare (416x240) at 60fps are tested. For each sequence, the first 50 frames are coded; only the first frame is coded as I-frame and the others are coded as P-frames. The experimental conditions related to H.264 are summarized in Table 1.

Our proposed PPM prediction is used only for coding the luminance component. In our PPM, the patch size n (as shown in Fig.1) is 5 and the block size m is 2. The overlapped region of patches in the PPM is one pixel. The number of reference frames is 4 and in each reference frame, 16 candidate patches are selected for generating the predicted block. In other words, K and I in (7) are 4 and 16, respectively. The PPM mode is selectively utilized at sub-block level according to (8).

Fig. 4 shows the coding performance of our proposed PPM in comparison with that of H.264/AVC. In this figure, results of our scheme and H.264/AVC are marked by "PPM" and "H.264" respectively. Fig.4 (a) presents the coding results of Foreman sequence. Improved coding performance can be achieved by our scheme at high bit rates. Compared with H.264, our scheme outperforms H.264 0.2dB in terms of PSNR. Fig. 4(b) further clarifies the coding gain of our scheme

at low bit rates. It can be observed that our PPM scheme constantly contributes from low to high bit rates. For BQSquare, similar coding gain is achieved by our scheme as shown in Fig (c) and (d).

The percentage of PPM usage is illustrated in Fig.5 and Fig.6. Fig.5 shows the result of the 2nd frame of Foreman sequence. Given the original frame (shown in Fig.5 (a)), the blocks coded using PPM mode are denoted by the black squares, as shown in Fig.5 (b). Similarly, Fig.6 presents the PPM usage of the 2nd frame of BQSquare. It can be seen that our PPM mode works for different kinds of blocks, including smooth blocks, texture blocks and edge blocks. For Foreman sequences, our experimental results show that the percentage of PPM mode usage ranges from 25% to 33% from high to low bit rate.

The corresponding residue images of the 2nd frames of Foreman and BQSquare are depicted in Fig.7 and Fig.8. Fig.7(a) and Fig.8(a) show the residues for H.264 prediction and Fig.7(b) and Fig.8(b) show the residues for the proposed PPM prediction. It can be observed that the predictions of our method are smoother and smaller than those of H.264 at the PPM coded regions.

Table 1. Test conditions related to H.264/AVC.

Item	*Parameter*
Number of reference frames	4
Search range	32
Entropy coding	CABAC
Transform 8x8 mode	Disable
Rate-distortion optimization	Enable
Quantization parameters	37, 32, 27, 22

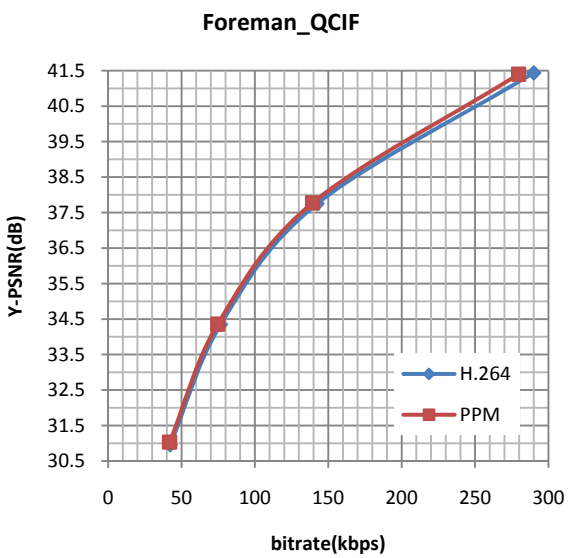
5. CONCLUSIONS

In this paper, we propose a predictive patch matching for inter prediction. It is inspired by the template matching in texture synthesis. By integrating the template matching with the traditional motion estimation, the predictive patch matching improves the efficiency of inter prediction. Furthermore, we extend the predictive patch matching into sub-block level. Experimental results demonstrate the effectiveness of our proposed predictive patch matching. Constant coding gain can be achieved by our scheme at both low and high bit rates.

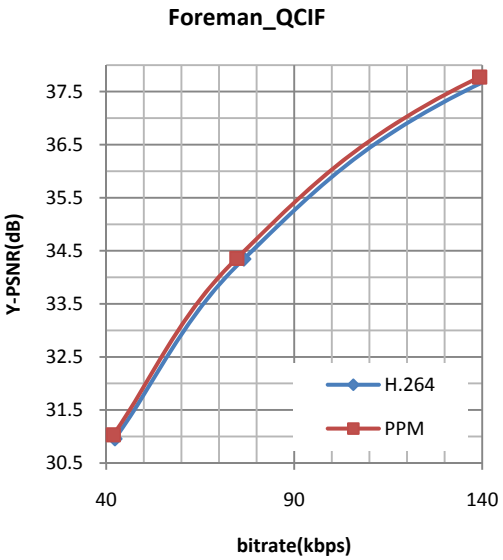
REFERENCES

1. F. Dufaux and F. Moscheni, “Motion estimation techniques for digital TV: a review and new contribution,” Proceeding of IEEE, Vol. 83, No.6, pp. 858-876, June 1995.
2. T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Trans on CSVT, vol. 13, no. 7, pp. 560–576, 2003.
3. A. Efros, and T. Leung, “Texture synthesis by non-parametric sampling,” in Proceedings of International Conference on Computer Vision, pp. 1033-1038, 1999.
4. S. Rane, G. Sapiro, and M. Bertalmio, “Structure and texture filling-in of missing image blocks in wireless transmission and compression applications,” IEEE Transactions on Image Processing, Vol.12, no. 3, pp. 296-303, Mar. 2003.
5. P. Ndjiki-Nya, T. Hinz, A. Smolic and T. Wiegand, “A generic and automatic content-based approach for improved H.264/MPEG-AVC video coding,” IEEE International Conference on Image Processing, vol. 2, pp. 874-877, Sept. 2005.
6. C. Zhu, X. Sun, F. Wu, and H. Li, “Video coding with spatio-temporal texture synthesis and edge-based inpainting,” Proceeding of ICME2008, pp813-816, 2008
7. Kazuo Sugimoto, Mitsuru Kobayashi, Yoshinori Suzuki, Sadaatsu Kato, and Choong Seng Boon, “Inter frame coding with template matching spatio-temporal prediction,” in Proc. IEEE Int. Conf. on Image Processing ICIP ’04, pp. 465–468, Oct. 2004

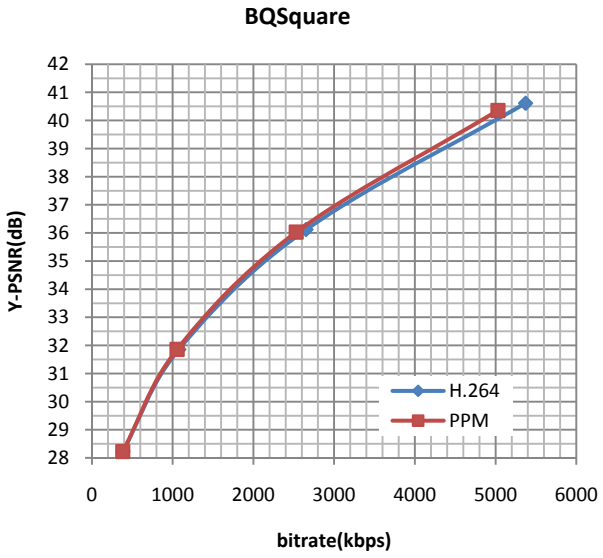
8. Yoshinori Suzuki, Choong Seng Boon, and Thiow Keng Tan, “Inter frame coding with template matching averaging,” in Proc. IEEE Int. Conf. on Image Processing ICIP ’07, San Antonio, TX, USA, Sept. 2007, pp. III–409–III–412.
9. S. Kamp, M. Evertz, and M. Wien, “Decoder side motion vector derivation for intra frame video coding,” in Proc. IEEE ICIP 2008, pp. 1120-1123, 2008
10. D. Marpe, H. Schwarz, and T.Wiegand, “Context-adaptive binary arithmetic coding in the H.264/AVC video compression standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 620–636, July 2003.
11. http://iphome.hhi.de/suehring/tml/download/



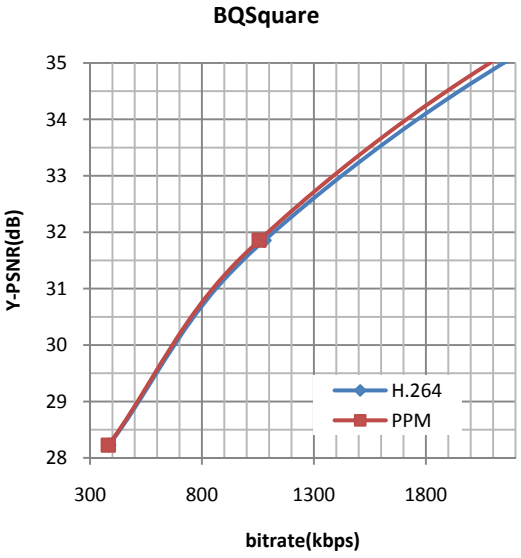
(a) foreman



(b) foreman at low bit rates



(c) BQSquare



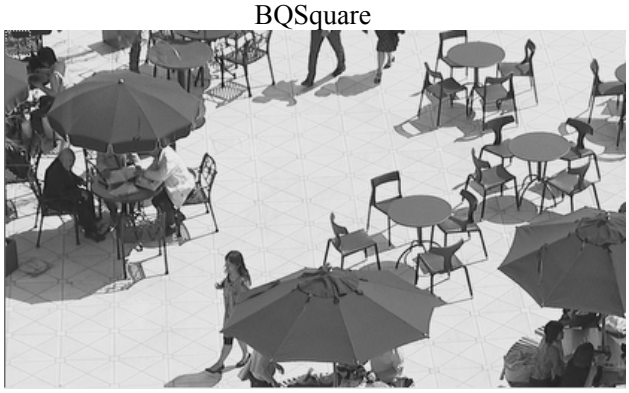
(d) BQSquare at low bit rates

Fig. 4. Coding performance evaluation of PPM in comparison with H.264. (a) Foreman at all bit rates; (b) Foreman at low bit rates; (c) BQSquare at all bit rates; and (d) BQSquare at low bit rates.

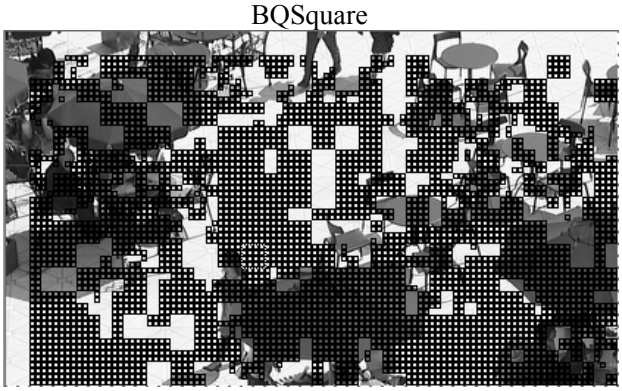
(a) Original frame size of 176x144.

(b) Black squares are blocks coded using PPM prediction mode.

Fig. **5**. PPM usage of the 2nd frame of Foreman at qp = 22. (a) Original frame; (b) MBs coded in PPM mode.

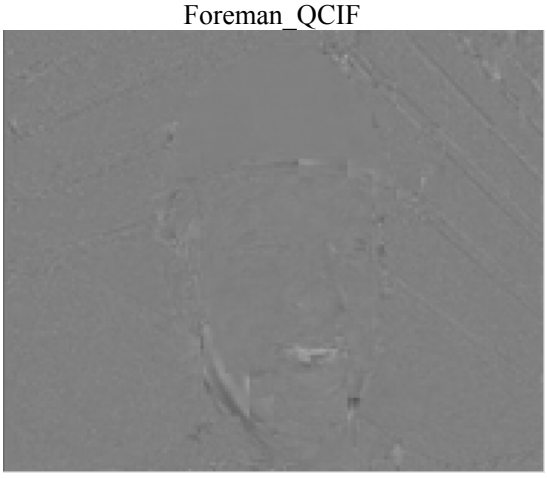


(a) Original frame size of 416x240

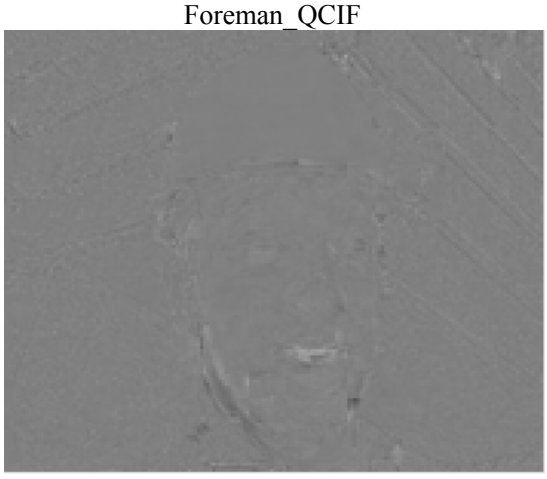


(b) Black squares are blocks coded using PPM prediction mode.

Fig. **6**. PPM usage of the 2nd frame of BQSquare at qp = 22. (a) Original frame; (b) MBs coded in PPM mode.

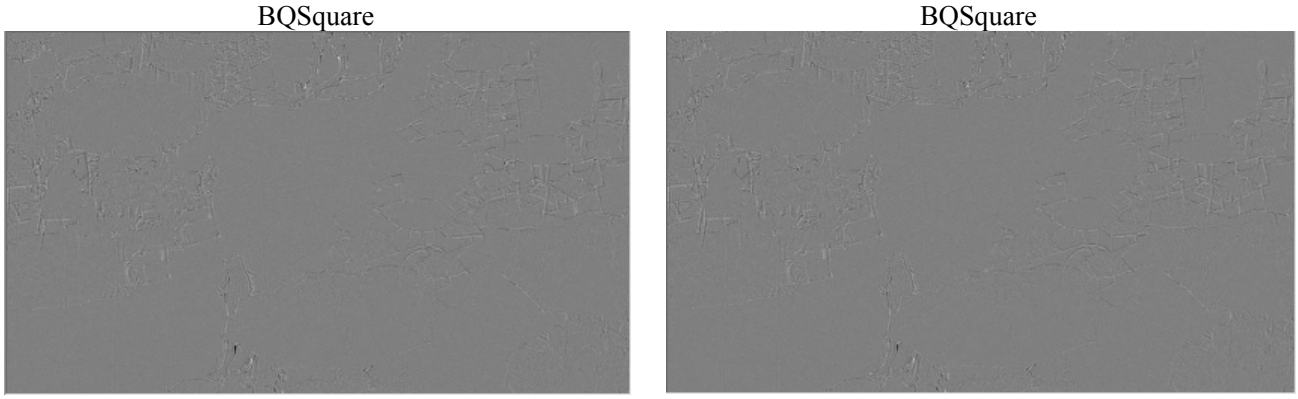


(a) Residues of H.264 prediction



(b) Residues of the proposed PPM prediction

Fig. **7**. Residues of the 2nd frame of Foreman at qp = 22. (a) Residues of H.264 prediction; (b) Residues of the proposed PPM prediction.



(a) Residues of H.264 prediction (b) Residues of the proposed PPM prediction

Fig. **8**. Residues of the 2nd frame of BQSquare at qp = 22. (a) Residues of H.264 prediction; (b) Residues of the proposed PPM prediction.