Parametric OBMC for Pixel-Adaptive Temporal Prediction on Irregular Motion Sampling Grids

Yi-Wen Chen and Wen-Hsiao Peng

***Abstract*—This paper adapts overlapped block motion compensation (OBMC) to suit variable block-size motion partitioning. The motion vectors (MVs) for various partitions are formalized as motion samples taken on an irregular grid. From this viewpoint, determining OBMC weights to associate with these samples becomes an under-determined problem since a distinct solution has to be sought for each prediction pixel. In this paper, we tackle this problem by expressing the optimal weights in closed form based on parametric signal assumptions. In particular, the computation of this solution requires only the geometric relations between the prediction pixel and its nearby block centers, leading to a generic framework capable of reconstructing temporal predictors from any irregularly sampled MVs. A modified implementation is also proposed to address the MV location uncertainty and to reduce computational complexity. Experimental results demonstrate that our scheme performs better than similar previous works, and when compared to the recently proposed Quadtree-based adaptive loop filter and enhanced adaptive interpolation filter, show a comparable gain. Furthermore, the combination of it with either of them gives a combined effect that is almost the sum of their separate improvements.**

***Index Terms*—Overlapped block motion compensation (OBMC), parametric window design, variable block size motion compensation (VBSMC), video coding.**

I. Introduction

BLOCK-BASED motion compensation (BMC) has been the most popular approach, in hybrid video coding schemes, for removing temporal redundancy. Conceptually, it uses one single motion vector (MV) as an estimate of the true motion field for a block of pixels, in order to trade off the accuracy of motion representation for less overhead. When chosen to minimize the mean squared block matching error, the MV is shown to approximate the true motion of the block center [19], [23]. It then makes good sense to view the function of BMC as a motion interpolator implementing a rectangular filter function. Obviously, such a crude interpolator can easily become problematic.

In the past, various algorithms have been proposed to improve BMC. The most straightforward technique is variable block-size motion compensation (VBSMC), which increases motion sampling density in areas with complex motion to compensate for the inefficiency of BMC. By contrast, control-grid interpolation (CGI) [17] and overlapped block motion compensation (OBMC) [16] use more sophisticated algorithms to reconstruct the motion field without additional samples. The former improves motion interpolation by employing a triangular filter function, while the latter directly gives a linear estimate of each pixel's intensity based on predictors derived from the current and nearby block MVs. Both are able to alleviate blocking artifacts effectively, but in practice, OBMC is preferred to CGI since the averaging of predictors also helps to reduce quantization noise [20]. To address the non-stationarity of the motion field, there are also hybrid schemes, which provide various combinations of BMC, CGI, and OBMC [6], [8], [9], [12]. However, they all require MVs to be sampled on a regular grid, i.e., to be generated by fixed block-size motion estimation.

Motivated by the preceding observations, we are led to seek an optimized hybrid of VBSMC and OBMC, aiming to trade better prediction for fewer MVs while retaining the flexibility to adapt motion sampling structure according to variations in image statistics. However, determining OBMC weights to associate with MVs on an irregular grid poses a challenging problem. This is because the variable block-size partitioning yields spatially varying geometric relations between a prediction pixel and its nearby block centers. In this case, solving for the weights with the least-squares method would become an under-determined problem since a distinct solution has to be sought for each possible context. Clearly, there may be more parameters to be estimated than there are data points.

This problem is not new. A similar situation occurred in the development of H.263 [1]. At that time, it was resolved by treating larger blocks as a collection of smaller blocks with the same MV in each smaller block as in the larger aggregate block and by applying a fixed window function to all MVs. In an attempt to extend the notion to H.264/AVC, Wang *et al.* [21] additionally proposed to weight more heavily those MVs from smaller aggregate blocks, which they believed can more reliably represent the motion of neighboring blocks, although no justification was given. Both methods suffer from the same problem that inner pixels in larger blocks are not properly compensated. Essentially, the MVs utilized for OBMC of those pixels are replicated from the same (aggregate) block MVs, producing a net effect like BMC. A third method that has recently been proposed is irregular-grid OBMC [6], which

Manuscript received October 30, 2010; revised February 28, 2011; accepted May 10, 2011. Date of publication May 31, 2011; date of current version January 6, 2012. This work was supported by the NSC, Taiwan, under NSC Project 100-2221-E-009-073. This paper was recommended by Associate Editor W. Gao.

The authors are with the Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: ewchen@csie.nctu.edu.tw; pawn@mail.si2lab.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2011.2158341

circumvents this deficiency by an adaptive window support that scales with local motion sampling density. It, however, remains unclear how to choose a proper scaling factor for each MV.

This paper departs from heuristic methods to approach the problem from a theoretical perspective. We formalize the notion of motion-compensated prediction (MCP) as a two-stage process consisting of sparse motion sampling followed by the reconstruction of temporal predictors. Within such a framework, OBMC in its generalized form is seen to find a linear minimum mean square error (LMMSE) estimate for every pixel's intensity based on motion-compensated signals derived from MVs sampled at nearby block centers. This viewpoint allows us to derive a parametric solution, termed parametric overlapped block motion compensation (POBMC), for determining the optimal weights in closed form. In doing so, the signal models in [23] are adopted to describe the probabilistic structures of the underlying intensity and motion fields. One important result of our POBMC is that its parameters include only the ℓ^2 distances between the locations of the prediction pixel and the MVs involved, i.e., their geometric relations are all that are needed to determine the weights. This leads to a generic method of reconstructing temporal predictors from any sparsely and irregularly sampled motion data.

Although our approach has some parallels with the other parametric solution [19], the unique features that distinguish this paper from it include the following.

1) Our focus is to adapt OBMC to suit variable block-size motion partitioning, while [19] concentrates on adjusting OBMC windows, based on the use of fixed block-size partitioning, in response to variations in sequence statistics.
2) We adopt an alternative signal model [23], which not only better represents the reality but also gives a result that is considerably more intuitive and tractable.
3) We address the uncertainty associated with a block MV's location by introducing a compensation term to reflect its dispersion around the block center.
4) We propose a suboptimal yet computationally efficient implementation, which need not solve the Wiener-Hopf equation and thus eliminates the need to compute matrix inverse.

In addition, we implement the proposed scheme with KTA 2.4r1 [11] and provide a performance comparison with the recently proposed enhanced adaptive interpolation filter (EAIF) [22] and Quadtree-based adaptive loop filter (QALF) [7] together with an analysis on how they interact with each other.

In the common test conditions, our POBMC delivers better rate-distortion (R-D) performance than both the H.263 OBMC [1] and the parametric solution [19]. Relative to an H.264/AVC anchor with extended macroblock (MB) size, it achieves 3.1% (0.7–13.6%) BD-rate reductions, compared to 4.6% (0.5–10.1%) and 7.2% (1.3–18.0%) with the single use of EAIF and of QALF, respectively. Although POBMC has the least gain among these filters, it can be combined efficiently with either of the other two filters. The result is an improvement that is almost the sum of their separate effects. In particular, the combination of POBMC and QALF performs very close to or better than that of EAIF and QALF, even in cases where the single use of EAIF outperforms that of POBMC.

The rest of this paper is organized as follows. Section II revisits the notion of MCP from a perspective based on motion sampling and reconstruction. Section III presents in detail the derivation of our parametric solutions. Section IV examines their properties by contrasting theoretical predictions with empirical data. Section V evaluates the compression performance of POBMC from various aspects and provides a runtime analysis. Section VI concludes this paper with a summary of our observations and a list of future works. Finally, the implementation details of POBMC are elaborated on in the Appendix.

II. Interpretation of MCP as Motion Sampling and Reconstruction

An insightful perspective on MCP is to view its process as consisting of sparse motion sampling followed by the reconstruction of temporal predictors. In this context, block-based motion estimation acts as a motion sampler taking samples at block centers while BMC interpolates, using the nearest-neighbor rule, between motion samples to construct the motion field. This interpretation facilitates a better understanding of various MCP schemes from a unified framework. For example, if we take such a view, VBSMC is merely an enhancement of BMC in motion sampling. The various MB partitionings are assimilated to different sampling structures, and choosing a specific block partitioning can be thought of as determining a local sampling pattern. By a similar reasoning, the difference between BMC and CGI is easily seen to be a different choice of motion interpolator. Somewhat less intuitive is OBMC, which does not directly reconstruct the motion field. Nevertheless, it was shown in [20] that an optimal OBMC window is also an optimal motion interpolation function, with which CGI can achieve the same mean squared prediction error as OBMC. This result furnishes another view of OBMC from the standpoint of motion interpolation. As an illustration, Fig. 1 contrasts graphically these techniques for the 1-D case.

With these ideas in mind, we cast the problem of combining OBMC with variable block-size motion partitioning as a linear estimate of a pixel's intensity based on predictors derived from motion samples taken on an irregular grid. The estimator will have the form of an finite impulse response filter with spatially varying coefficients due to the variations in local motion sampling structure. Particularly, we shall first assume that all motion samples are positioned exactly at block centers. Then, the derivation will be repeated to consider the uncertainty of their locations owing to the use of variable block-size motion search. Developing these methods is the main contribution of this paper and we shall in the following discuss each of them in detail.

III. POBMC

A. Review of OBMC

This section briefly reviews the basics of OBMC, to aid the understanding of our POBMC. In words, OBMC is to find a

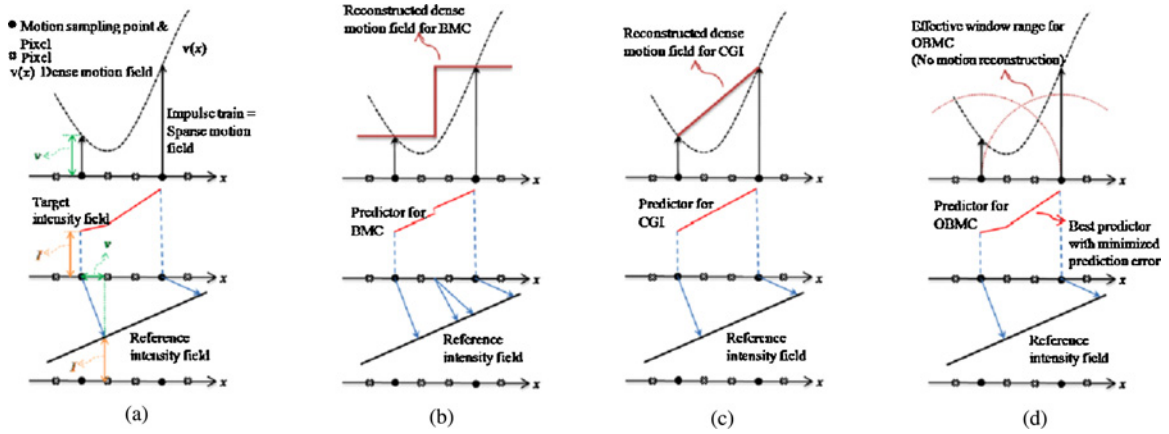


Fig. 1. Various MCP schemes in the 1-D case. (a) MCP based on the true motion field. (b) BMC. (c) CGI. (d) OBMC.

LMMSE estimate of a pixel's intensity value $I_k(\mathbf{s})$ based on motion-compensated signals $\{I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))\}_{i=1}^{L}$ derived from its nearby block MVs $\{\mathbf{v}(\mathbf{s}_i)\}_{i=1}^{L}$. From an estimation-theoretic perspective, these MVs are plausible hypotheses for its true motion, and to maximize coding efficiency, their weights $\mathbf{w} = [w_1, w_2, \ldots, w_L]^T$ are chosen to minimize the mean squared prediction error subject to the unit-gain constraint [16] as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \xi(\mathbf{w}) \text{ s.t. } \sum_{i=1}^{L} w_i = 1 \tag{1}$$

where

$$\xi(\mathbf{w}) = E\left\{\left(I_k(\mathbf{s}) - \sum_{i=1}^{L} w_i I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))\right)^2\right\}.$$

Applying the Lagrangian method to (1) then gives

$$\mathbf{w}^* = \mathbf{R}^{-1}\left[\mathbf{P} - \mathbf{U}\left(\frac{\mathbf{U}^T\mathbf{R}^{-1}\mathbf{P} - 1}{\mathbf{U}^T\mathbf{R}^{-1}\mathbf{U}}\right)\right] \tag{2}$$

where $[\mathbf{R}]_{ij} = E[I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j))]$ and $[\mathbf{P}]_j = E[I_k(\mathbf{s})I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j))]$ stand, respectively, for auto-correlation and cross-correlation matrices, and $\mathbf{U}$ is a column vector with all elements equal to one [16]. Given that the underlying intensity and motion fields are stationary and that motion samples are taken on a square lattice (such is the case when an image is divided into a group of square blocks for motion search), the optimal weights $\mathbf{w}^*$ for pixel $\mathbf{s}$ depend solely on its relative position within a block. They are often obtained using the least-squares method due to lack of knowledge of the probabilistic models of real data.

The concept of OBMC can be generalized to the case where motion sampling structure is irregular. The challenge, however, becomes how to compute for each pixel its optimal weights to associate with nearby MVs, given that both auto-correlation and cross-correlation functions are spatially varying. The least-squares solution, although feasible in theory, is impractical because the storage of weighting coefficients optimized for different contexts demands huge memory requirements. To tackle this problem, we resort to a parametric solution.

B. Signal Models

POBMC aims to give a closed-form formula for the optimal weights. To do so, it usually needs to assume signal models for the intensity and motion fields. The choice of the models often involves a tradeoff between accuracy, simplicity, and tractability, and can sometimes be quite subtle. For instance, Tao *et al.* [19] model the auto-correlation functions of the intensity and motion fields using quadratic and exponential functions, respectively. These models are so chosen that $\mathbf{R}$ and $\mathbf{P}$ can be expressed in closed form. In general, different models have their merits and faults, and what model best represents reality is normally justified by empirical simulations.

In this paper, we aim to give a direct estimate of the optimal weights $\mathbf{w}^*$. This is accomplished by adopting the motion model proposed in [23], which assumes that the difference between the true motion of any two pixels, e.g., $\mathbf{s}_1$ and $\mathbf{s}_2$, has a normal distribution of the form

$$v_x(\mathbf{s}_1) - v_x(\mathbf{s}_2) \text{ or } v_y(\mathbf{s}_1) - v_y(\mathbf{s}_2) \sim \mathcal{N}(0, \alpha r^2(\mathbf{s}_1, \mathbf{s}_2)) \tag{3}$$

where α is a positive number indicating the degree of motion randomness in the horizontal or vertical direction,[1] and $r(\mathbf{s}_1, \mathbf{s}_2)$ is the ℓ^2 distance (measured in the unit of pixel) between $\mathbf{s}_1$ and $\mathbf{s}_2$. Caution, however, must be exercised when using (3) because it is an incomplete specification. The variance $\alpha r^2(\mathbf{s}_1, \mathbf{s}_2)$ must be bounded from above for the model to be proper. To see this, let us assume the motion field is stationary and symmetric. It then follows from (3) that

$$E\{v_x(\mathbf{s}_1)v_x(\mathbf{s}_2)\} = E\{v_y(\mathbf{s}_1)v_y(\mathbf{s}_2)\} = \sigma_m^2 + \mu_m^2 - \frac{\alpha r^2(\mathbf{s}_1, \mathbf{s}_2)}{2} \tag{4}$$

where μ_m and σ_m^2 are the mean and the variance of the motion field, respectively. Using the Cauchy-Schwarz inequality, we have $4\sigma_m^2 \geq \alpha r^2(\mathbf{s}_1, \mathbf{s}_2) \geq 0$. The lower bound is obvious, but the upper bound deserves more attention. According to (4), it implies that the MVs of two far-away pixels are negatively correlated. A tighter bound that agrees more with the general observation is $2\sigma_m^2$, which will make them become uncorrelated. We can equivalently define a clipper function for

[1]The smaller α value suggests the motion field has higher spatial correlation.

$r(\mathbf{s}_1, \mathbf{s}_2)$ to have the property $\widehat{r}(\mathbf{s}_1, \mathbf{s}_2) = Clip(0, r(\mathbf{s}_1, \mathbf{s}_2), \tau)$, where the clipping threshold $\tau = \sqrt{2\sigma_m^2/\alpha}$. Hereafter, we shall omit the tedious repetition of this constraint by using $\widehat{r}(\mathbf{s}_1, \mathbf{s}_2)$ in place of $r(\mathbf{s}_1, \mathbf{s}_2)$.

C. Optimal Weights in Parametric Form

With the signal model in (3), we next proceed to determine the optimal weights $\mathbf{w}^*$ using *calculus*. To begin with, we rewrite, by noting that $\sum_{i=1}^{L} w_i = 1$, the mean squared prediction error $\xi(\mathbf{w})$ in (1) as

$$\xi(\mathbf{w}) = E\left\{\left(\sum_{i=1}^{L} w_i d(\mathbf{s}; \mathbf{v}(\mathbf{s}_i))\right)^2\right\} \tag{5}$$

where $d(\mathbf{s}; \mathbf{v}(\mathbf{s}_i)) = I_k(\mathbf{s}) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_i))$ denotes the residual signal when $I_k(\mathbf{s})$ is predicted from the motion-compensated signal $I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_i))$ using the MV $\mathbf{v}(\mathbf{s}_i)$ for block i. Equation (5) can be written more compactly in matrix notation as

$$\xi(\mathbf{w}) = \mathbf{w}^T E\{\mathbf{d}\mathbf{d}^T\}\mathbf{w} = \mathbf{w}^T \mathbf{D}\mathbf{w} \tag{6}$$

where $\mathbf{d} = [d(\mathbf{s}; \mathbf{v}(\mathbf{s}_1)), d(\mathbf{s}; \mathbf{v}(\mathbf{s}_2)), \ldots, d(\mathbf{s}; \mathbf{v}(\mathbf{s}_L))]^T$.

To continue, we borrow a result in [23], which shows that if (3) is valid, then $E\{d^2(\mathbf{s}; \mathbf{v}(\mathbf{s}_i)\}$ has a closed-form formula given by

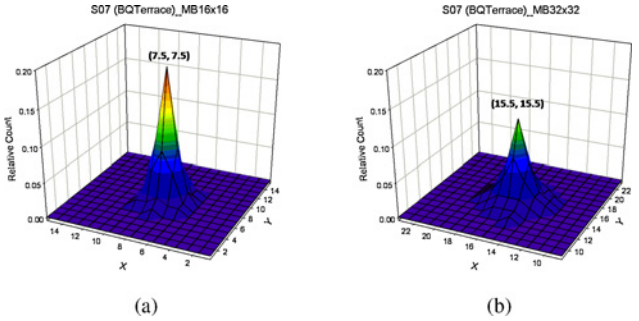
$$\begin{aligned} E\{d^2(\mathbf{s}; \mathbf{v}(\mathbf{s}_i)\} &= E\{(I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s})) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_i)))^2\} \\ &= \epsilon \widehat{r}^2(\mathbf{s}, \mathbf{s}_i) \end{aligned} \tag{7}$$

where ϵ is a constant indicating the joint randomness of the motion and intensity fields, $I_k(\mathbf{s}) = I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}))$ with $\mathbf{v}(\mathbf{s})$ denoting the true motion of pixel $\mathbf{s}$, and the block MV $\mathbf{v}(\mathbf{s}_i)$ is approximated as the motion associated with the block center $\mathbf{s}_i$. What remain to be determined in $\mathbf{D}$ are those off-diagonal entries, i.e., $E\{d(\mathbf{s}; \mathbf{v}(\mathbf{s}_i)d(\mathbf{s}; \mathbf{v}(\mathbf{s}_j)\}, i \neq j$; in fact, their derivations are merely an application of (7). With a little bit of algebra,[2] we obtain

$$\begin{aligned} & E\{d(\mathbf{s}; \mathbf{v}(\mathbf{s}_i)d(\mathbf{s}; \mathbf{v}(\mathbf{s}_j)\} \\ = \;& E\{(I_k(\mathbf{s}) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_i)))(I_k(\mathbf{s}) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_j)))\} \\ = \;& \frac{1}{2}E\{(I_k(\mathbf{s}) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_i))^2\} \\ & + \frac{1}{2}E\{(I_k(\mathbf{s}) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_j))^2\} \\ & - \frac{1}{2}E\{(I_k(\mathbf{s} + \mathbf{v}(\mathbf{s}_i)) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_j))^2\} \\ = \;& \frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s}, \mathbf{s}_i) + \widehat{r}^2(\mathbf{s}, \mathbf{s}_j) - \widehat{r}^2(\mathbf{s}_i, \mathbf{s}_j)\right). \end{aligned} \tag{8}$$

The astute reader may feel a sense of misgiving about the approximation $E\{(I_k(\mathbf{s} + \mathbf{v}(\mathbf{s}_i)) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_j))^2\} \approx \epsilon \widehat{r}^2(\mathbf{s}_i, \mathbf{s}_j)$, as it does not seem to be a direct extension of (7). The subtle difference is the replacement of $\mathbf{v}(\mathbf{s})$ with $\mathbf{v}(\mathbf{s}_i)$. However, assuming that $\mathbf{v}(\mathbf{s}_i)$ represents the true motion of the block center $\mathbf{s}_i$, its proof can be carried out in the same manner as for (7). Another testament to its mathematical correctness is that (9) includes (7) as a special case where $\mathbf{s}_i = \mathbf{s}_j$.

[2] $(a-b)(a-c) = \frac{1}{2}(a-b)^2 + \frac{1}{2}(a-c)^2 - \frac{1}{2}(b-c)^2$.



(a) (b)

Fig. 2. Distribution of a block MV's location when the block size used for motion search is varied. (a) 16×16. (b) 32×32. The MV location is approximated by the centroid position of the first ten pixels, in a block, having relatively smaller prediction error.

Returning to (6), we are now ready to find the optimal weights. Since $\xi(\mathbf{w})$ is to be minimized subject to $\sum_{i=1}^{L} w_i = 1$, the solution space has only a dimension of $L-1$. To simplify the computation, we define a reduced-dimension weight vector $\widetilde{\mathbf{w}} = [\widetilde{w}_1, \widetilde{w}_2, \ldots, \widetilde{w}_{L-1}]^T$, the elements of which are free variables and are related to the weight vector $\mathbf{w}$ by

$$\mathbf{w} = \mathbf{e} - \mathbf{M}\widetilde{\mathbf{w}} \tag{9}$$

where

$$\mathbf{e} = [0, 0, \ldots, 1]^T_{L\times 1} \text{ and}$$

$$\mathbf{M} = \begin{bmatrix} -\mathbf{I} \\ \mathbf{U}^T \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}_{L\times(L-1)}.$$

When spelled out, (9) simply states that $w_i = \widetilde{w}_i$, $1 \le i \le L-1$ and $w_L = 1 - \sum_{i=1}^{L-1} \widetilde{w}_i$. Substituting (9) into (6), setting the gradient of ξ with respect to $\widetilde{\mathbf{w}}$ to $\mathbf{0}$, and solving the resulting system equations then yields

$$\widetilde{\mathbf{w}}^* = (\mathbf{M}^\mathbf{T}\mathbf{D}\mathbf{M})^{-1}\mathbf{M}^\mathbf{T}\mathbf{D}\mathbf{e}. \tag{10}$$

The result of $\widetilde{\mathbf{w}}^*$ immediately gives that of $\mathbf{w}^*$ by (9) as follows:

$$\mathbf{w}^* = \left(\mathbf{I} - \mathbf{M}(\mathbf{M}^\mathbf{T}\mathbf{D}\mathbf{M})^{-1}\mathbf{M}^\mathbf{T}\mathbf{D}\right)\mathbf{e}. \tag{11}$$

Inspection of (11) reveals that the optimal weights depend solely on the distances between the prediction pixel $\mathbf{s}$ and the block centers involved $\{\mathbf{s}_i\}_{i=1}^{L}$. The ϵ term is absent in the final result. This remarkable property allows MVs sampled on a possibly irregular grid to be incorporated for OBMC, providing a reconstruction method applicable to any sampling structures.

D. Optimal Weights in a Special Case

An interesting special case occurs by considering $\mathbf{D}$ as a diagonal matrix. In this case, the prediction errors $\{d(\mathbf{s}; \mathbf{v}(\mathbf{s}_i)\}_{i=1}^{L}$

are uncorrelated with each other, i.e., $E\{d(\mathbf{s};\mathbf{v}(\mathbf{s}_i)d(\mathbf{s};\mathbf{v}(\mathbf{s}_j)\} = 0, \forall i \neq j$, and $\mathbf{w}^*$ becomes

$$\mathbf{w}^* = \left(\sum_{i=1}^{L}\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_i)}\right)^{-1}\left[\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)},\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_2)},\ldots,\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_L)}\right]^T. \tag{12}$$

The proof of this result requires some work but involves only straightforward computations. Equation (12) is a great simplification of (11); the optimal weights w_i^* are simply the normalized inverses of the corresponding squared distances between $\mathbf{s}$ and $\mathbf{s}_i$. It has the interpretation that prior to normalization, the contribution of each MV $\mathbf{v}(\mathbf{s}_i)$ to estimating its nearby pixel intensities is a function of pixel $\mathbf{s}$ that decays quadratically with $\widehat{r}(\mathbf{s},\mathbf{s}_i)$. If we take such a view, other functions can be substituted for $1/\widehat{r}^2(\mathbf{s},\mathbf{s}_i)$. For example, it may be just as well to adopt the raised cosine or bilinear function of various supports, or to change the power of $1/\widehat{r}(\mathbf{s},\mathbf{s}_i)$. As an afterthought, each of these functions may correspond to making some specific assumptions about the motion and intensity fields. Due to its simplicity, (12) will be included in the following sections as an alternative to (11).

E. MV Location Uncertainty

In the preceding derivation, we have always assumed that a block MV represents the true motion of the block center. However, this is an approximation; in fact, it may correspond to the motion of any pixel around the center. To see this, consider a small group of pixel locations in a block where prediction errors are relatively smaller. We think of the block MV as the motion connected to their centroid. Although not precise, this expedient provides a rough estimate of the MV location without having to acquire the true motion field. Fig. 2 presents two plots showing the centroid distributions when the block size used for motion search is varied. Two observations are immediate: 1) the means of both distributions are close to the block center, which justifies the widely accepted approximation, and 2) the variance is nonzero and increases with the increasing block size, which suggests that the locations of $\mathbf{s}_i, \mathbf{s}_j$ in (7) and (8) should be modeled probabilistically.

We now generalize both equations to consider their random effects. To conform with our previous notation, we denote by $\widetilde{\mathbf{s}}_i = \mathbf{s}_i + \mathbf{n}_i$ (respectively, $\widetilde{\mathbf{s}}_j = \mathbf{s}_j + \mathbf{n}_j$) their true locations, which are characterized by an independent, additive noise vector $\mathbf{n}_i$ (respectively, $\mathbf{n}_j$) with mean zero and covariance matrix as follows:

$$\mathbf{K}_{\mathbf{n}_i\mathbf{n}_i} = \begin{bmatrix} \delta_i^{(x)} & \rho_i\sqrt{\delta_i^{(x)}\delta_i^{(y)}} \\ \rho_i\sqrt{\delta_i^{(x)}\delta_i^{(y)}} & \delta_i^{(y)} \end{bmatrix}.$$

Substituting $\widetilde{\mathbf{s}}_i$ for $\mathbf{s}_i$ in (7) and applying the law of iterated expectations, we get

$$\begin{aligned} & E\{d^2(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_i)\} \\ = \;& E\left\{E\{d^2(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_i))|\widetilde{\mathbf{s}}_i\}\right\} = \epsilon E\left\{\widehat{r}^2(\mathbf{s},\widetilde{\mathbf{s}}_i)\right\} \\ \simeq \;& \epsilon E\left\{(\mathbf{s}^{(x)} - \mathbf{s}_i^{(x)} - \mathbf{n}_i^{(x)})^2 + (\mathbf{s}^{(y)} - \mathbf{s}_i^{(y)} - \mathbf{n}_i^{(y)})^2\right\} \\ \simeq \;& \epsilon\widehat{r}^2(\mathbf{s},\mathbf{s}_i) + \epsilon(\delta_i^{(x)} + \delta_i^{(y)}) \end{aligned} \tag{13}$$

where the superscripts x, y indicate the two components of a point or a vector. In (13), the locations of pixel $\mathbf{s}$ and the block center $\mathbf{s}_i$ are treated as known variables because we know exactly what MVs will be utilized for the motion compensation of pixel $\mathbf{s}$. As such, they are deterministic quantities and the expectation in the penultimate approximation is taken with respect to $\mathbf{n}_i$ only. In the course, we have tacitly ignored the clipping effect on $r(\mathbf{s},\widetilde{\mathbf{s}}_i)$, which however is crucial for our signal models to be proper (Section III-B). A way out of this difficulty is to assume that $\mathbf{s}_i$ is close enough to $\mathbf{s}$ so that the result in (13) is a good approximation. This assumption can be justified to some extent since in practical implementation of our schemes, we use only those neighboring MVs that are closer to a pixel for its motion compensation. From (13), the consequence of MV location uncertainty is an increase in the mean squared prediction error. Of particular interest is that the penalty depends only on the variances of $\mathbf{n}_i$ (or equivalently, the variances of $\widetilde{\mathbf{s}}_i$) regardless of its distribution.

A similar calculation leads us to

$$\begin{aligned} & E\{d(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_i)d(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_j)\} \\ = \;& \frac{1}{2}\epsilon E\left\{\left(\widehat{r}^2(\mathbf{s},\widetilde{\mathbf{s}}_i) + \widehat{r}^2(\mathbf{s},\widetilde{\mathbf{s}}_j) - \widehat{r}^2(\widetilde{\mathbf{s}}_i,\widetilde{\mathbf{s}}_j)\right)\right\} \\ \simeq \;& \frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s},\mathbf{s}_i) + \delta_i^{(x)} + \delta_i^{(y)}\right) + \frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s},\mathbf{s}_j) + \delta_j^{(x)} + \delta_j^{(y)}\right) \\ - \;& \frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s}_i,\mathbf{s}_j) + \delta_i^{(x)} + \delta_i^{(y)} + \delta_j^{(x)} + \delta_j^{(y)}\right) \\ = \;& \frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s},\mathbf{s}_i) + \widehat{r}^2(\mathbf{s},\mathbf{s}_j) - \widehat{r}^2(\mathbf{s}_i,\mathbf{s}_j)\right) \end{aligned}$$

where $\mathbf{n}_i$ and $\mathbf{n}_j$ are assumed to be independent. As shown, the variance terms in (14) cancel each other out, leading to the same result as in (8). Simply substituting (13) into the matrix $\mathbf{D}$ in (11) gives the *modified optimal weights* with consideration of MV location uncertainty. These results also apply to the case where $\mathbf{D}$ is a diagonal matrix.

In concluding this section, we want to point out that the proposed scheme has two parameters to be determined: the clipping threshold τ and the degree of MV location uncertainty $\delta = \delta_i^{(x)} + \delta_i^{(y)}$. The latter actually denotes a set of parameters, one for each distinct block size. As will be discussed later, they can be determined by offline training.

IV. Analysis of Window Functions

While (11) characterizes the contributions of a set of MVs to estimating the intensity of a pixel, an equivalent yet more insightful perspective is to see the window function of each MV, which specifies its weights used to estimate pixel intensities in a neighborhood [16]. In this section, we shall gain further insights into the proposed solutions from this viewpoint. To ease comprehension, we first consider the simpler case of fixed block-size motion partitioning, followed by the more sophisticated one involving variable block-size partitioning.

A. Theoretical Window Functions

Fig. 3(a)–(c) plots the window functions for various clipping threshold values τ's. Their counterparts in Fig. 3(d)–(f) shows the results when the off-diagonal entries of $\mathbf{D}$ are set zero. In

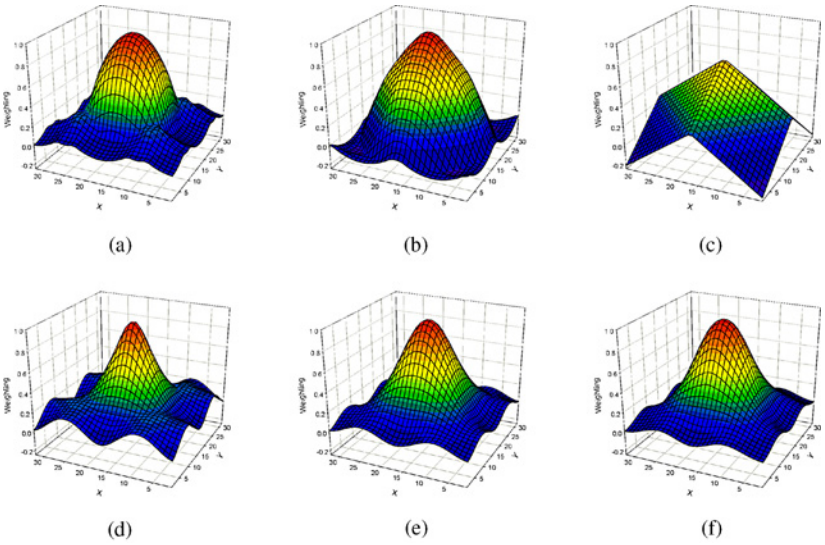


Fig. 3. Effect of the clipping threshold value on the shape of the proposed parametric windows with (a)–(c) non-diagonal and (d)–(f) diagonal D matrices. From left to right, the clipping threshold values are 10, 15, 35, respectively.

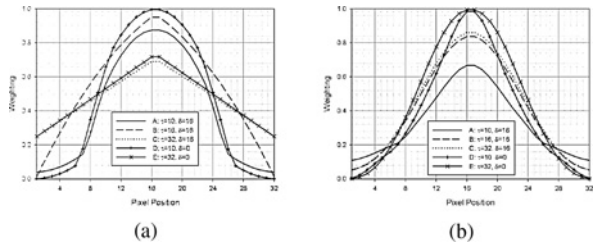


Fig. 4. Window functions along the slide of $Y = 16$ based on a (a) non-diagonal or (b) diagonal D matrix.

the former case, we observe that the window shape inflates with the increasing τ, and eventually converges to a bilinear function. This trend of inflation continues in the latter case although the change in the window shape is not that radical, especially when the value of τ becomes high enough. These phenomena can be explained by noting that a higher clipping threshold implies a stronger correlation between the motion of different pixels (smaller α) or a larger motion variance (larger σ_m^2). Under these circumstances, it is intuitive to expect that the influence of a block MV will extend to more pixels.

To gain a better appreciation of how the window shape evolves, Fig. 4 further displays the cross sections of these windows along the slide $Y = 16$. There are several points to be noted here. First, the weights around the block center ($X = 16$) are seen to be smaller than 1. This result is a manifestation of MV location uncertainty. As expected, their values tend to approach 1 if we have $\delta = \delta_i^{(x)} + \delta_i^{(y)} = 0$ [see (13)]. Some other interesting observations follow from comparing the window values at $X = 16.5$ (current block center) and at $X = 0.5$ or 32.5 (neighboring block centers). The windows with a diagonal **D** resemble normal functions in shape, and exhibit an upward trend in magnitude near the block center (respectively, a downward trend at the neighboring block centers) as the value of τ increases. In the general case, however, the behavior is more intricate: the peak value escalates first and then declines. But, both cases have one thing in common: their windows converge to a function dependent only on δ when τ is large enough.

B. Comparison with Empirical Window Functions

The different results above lead us to wonder which model is more reasonable and how much the penalty is for keeping only the diagonal entries of **D**. In this section, we provide empirical justifications by contrasting the parametric windows with those obtained by the least-squares method. Results of [19] are also included for comparison. Particularly, to demonstrate the best achievable performance of our parametric schemes, both the values of τ and δ are searched exhaustively based on minimizing the mean squared prediction error, and so is the parameter ρ_m in [19].[3]

From the results presented in Fig. 5(a), we see that the proposed windows with a non-diagonal **D** match closely the least-squares ones. The other windows, although showing similar magnitudes at block boundaries ($X = 8.5$ or 24.5), have much higher weights near the block center ($X = 16.5$). Despite their distinct appearances, the penalties in mean-squared error (MSE) are somehow surprisingly not as high as expected. We find that there are actually several window functions (of different shapes) performing almost equally well. To see this, Fig. 5(b) plots the resulting MSEs as functions of τ and δ for the case of non-diagonal **D**. The best settings, which yield similar MSEs as achieved by the optimal windows, are labeled "minimum." From the figure, these settings roughly have $\tau = 15 \sim 20$ and $\delta = 100 \sim 200$; however, many other choices deliver similar performance. For example, all the settings in Fig. 4, except setting B, are among the best performing ones even though their window shapes differ noticeably. In particular, it seems beneficial (in terms of MSE) to have a high clipping threshold. This may be attributed to the

[3]The parametric solution in [19] originally has four parameters to be determined. But, a little neat algebra shows that the resulting window is dependent on only the correlation coefficient of the motion field, ρ_m.

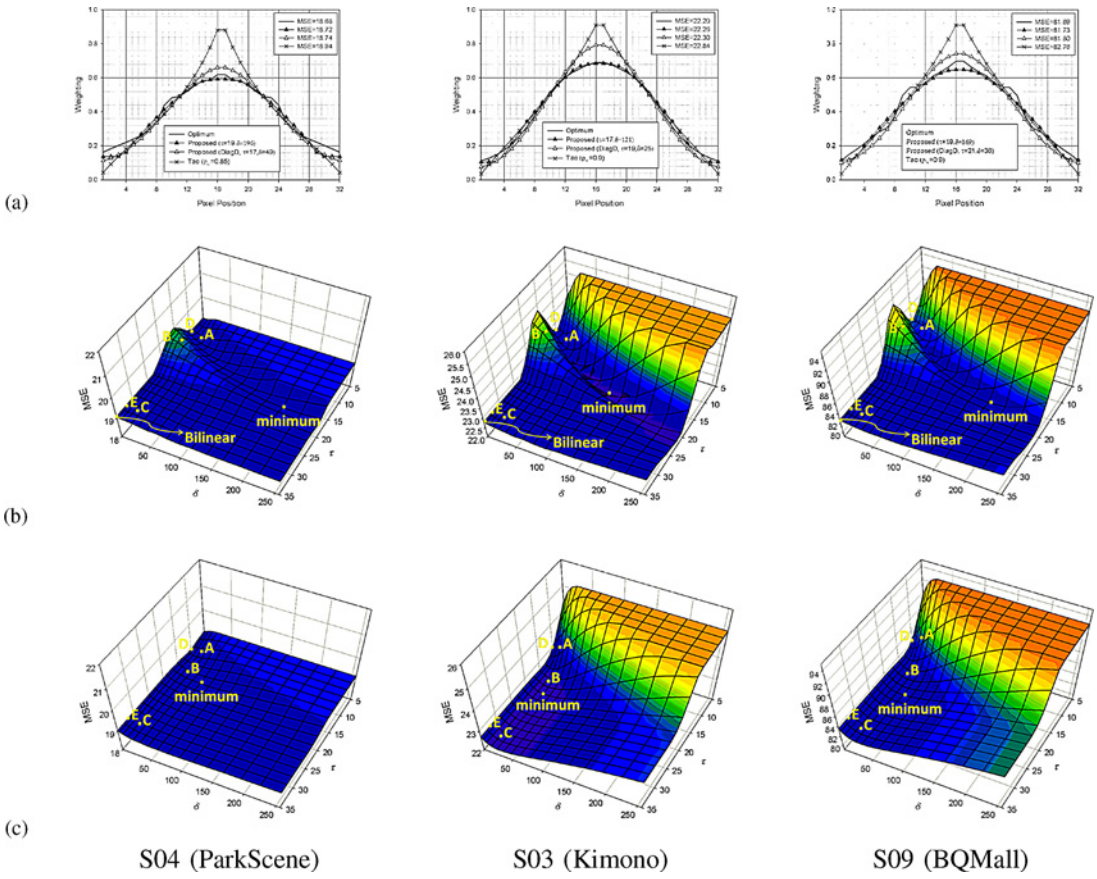


Fig. 5. Comparisons of window functions and their MSE surfaces. (a) Parametric windows versus optimal least-squares windows. The MSE surfaces of the proposed parametric solution with a (b) non-diagonal or (c) diagonal D matrix.

incorporation of an R-D based motion search criterion, which imposes a motion smoothness constraint and thus increases the motion correlation. Notice that the MSE stops decreasing when τ exceeds a certain value because, as mentioned previously, the window converges to a function dependent on δ. From Fig. 5(b), one may doubt the necessity of using parametric windows, since the bilinear function shown in Fig. 3(c) seems to provide sufficiently good performance. While this is indeed the case for fixed block-size motion partitioning, it is not true for the variable block-size case. As will be seen in the next section, difficulties arise when a fixed window is to be applied to MVs sampled on an irregular grid. In addition, it is worth noting that this bilinear function differs from the usual one [16] in that its peak value at the block center is much smaller than 1, which has to do with the MV location uncertainty. A side experiment shows that the latter yields larger MSEs due to the ignorance of this uncertainty. Finally, some sequences are seen to be more sensitive to parameter selection than others.

Fig. 5(c) further shows the results for the case of diagonal **D**. Comparing with Fig. 5(b), there is an obvious distinction in the region with $\tau \geq 15$, where the MSE first decreases slightly along the δ-axis and then increases rapidly (in some cases) with the increasing δ. The "minimum" points thus have $\delta = 10 \sim 50$. Analogous to the general case, there are also more than one window function showing similar performance to the least-squares solutions; examples are settings B and C in Fig. 4. Interestingly, what are improper settings in the general

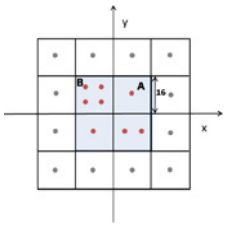


Fig. 6. Irregular motion sampling grid due to the use of variable block-size motion partitioning.

case now become proper ones. This is because the meanings of both parameters change when we twist **D**. It is for the same reason that the estimation of their values becomes difficult. Fortunately, their optimal values are found empirically to be less sequence-dependent, even if not truly independent, and can be obtained offline. According to the MSE results in Fig. 5(a), ignoring the off-diagonal entries of **D** does not seem to have a significant impact on prediction performance, if δ and τ are chosen properly. Because of its simplicity and fairly good performance, we shall hereafter adopt (12) as our parametric solution.

C. Window Functions on Irregular Sampling Grids

The discussion so far has been restricted to window functions for regular motion sampling structures. We now turn our attention to irregular ones, which could result from an

application of variable block-size motion partitioning. One example of such a structure is given in Fig. 6.

Fig. 7(a) and (b) plots the window functions associated with those MVs in the shaded area for the cases of non-diagonal and diagonal **D**, respectively. Some of them are displayed separately for close scrutiny. We see that these functions may not be symmetric, and their shapes depend highly on the local sampling pattern. An interesting observation is that windows in areas with a higher motion sampling density (i.e., in MBs with more partitions) are less concentrated; this implies a stronger averaging of the relevant MVs. In theory, it seems reasonable as these MVs, spatially closer to each other, are assumed to be highly correlated, but in practice, this may not always be the case. Sometimes a MB is segmented into smaller partitions because it spans across multiple objects with different motion. We thus propose to adjust the value of δ in a MB-adaptive manner. Recall that it indicates the dispersion of a MV's location around the block center; generally, MVs for smaller blocks should have a smaller δ value. After such adjustment, the results, as illustrated in Fig. 7(c), are more centralized windows in high density areas, which help to mitigate the over-blurring of block boundaries. In particular, the tuning of δ need not be fine-granular; an adaptive selection between two distinct levels is sufficient (Section V).

Finally, the results based on the window function in H.263 are shown in Fig. 7(d). Recall that in H.263, a fixed window function is used for OBMC. To apply the same window to all MVs, the issue of variable block-size motion partitioning is resolved by treating larger blocks as a collection of smaller blocks with the same MV in each smaller block as in the larger aggregate block. Hence, in computing the effective weighting factor of a MV at some specific pixel, we add up the contributions of all its replicas. From the figure, the window functions for larger blocks have a value equal or close to 1 around the block centers, which implies their inner pixels are not properly compensated by OBMC. This result is a direct consequence of the MV replication mechanism. As will be seen next, this approach leads to poor compression performance; in fact, applying any fixed window with the same approach will suffer from a similar problem.

V. Experimental Results

The proposed POBMC is analyzed by a series of tests: the first test compares its compression performance with that of similar previous works, including the OBMC in H.263 [1] and the parametric solution in [19]; the second test contrasts POBMC with the two in-loop filters, EAIF [22] and QALF [7], in KTA [11]; and the third test studies how these in-loop filters interact with each other in a complete codec. Finally, we conclude this section with a software runtime analysis, to offer a rough indication of its complexity characteristics. All OBMC schemes are implemented in KTA 2.4r1 [11], with details given in the Appendix. All tests, unless otherwise stated, use the configurations shown in Table I. The results are obtained by encoding the first 100 frames of standard JCT-VC test sequences [2]. Those for POBMC are produced using (12) with $\tau = 32$.

TABLE I
Encoder Configurations

Configuration	Setting		
GoP Structure	IPPP...	IBBB...	Hierarchical B
Intra period	0	0	24
QPP and QPB	QPI+1	QPI+1	QPI+1/2/3/4
QPI	22, 27, 32, 37		
CABAC	On		
Reference frames	4 (P), 2 (B_L0, B_L1)		
8×8 transform	On		
De-blocking	On		
RDO	On		
MV range	± 128		
Motion search	3 (EPZS)		
Sub-pixel MC	On		
Adaptive rounding	Off		
Motion comp. block sizes	8×8 to 32×32		

A. R-D Performance Analyses

1) *Comparison of OBMC Algorithms:* Table II compares the compression performance of different OBMC schemes by showing their BD-rate savings [4] relative to an anchor encoding, which conforms to H.264/AVC High Profile with inclusion of extended MB size (up to 32×32). In particular, they all involve an adaptive switching between BMC and one or more OBMC options. The subscript H-I indicates the use of only one OBMC option. For example, $\text{POBMC}_{\text{H-I}}$ denotes the hybrid of BMC and POBMC with $\delta = 16$, and $\text{POBMC}_{\text{H-II}}$ includes one more option with $\delta = 0$. To signal the choice of MCP schemes, a flag is sent for each non-skip super MB (a MB of size 32×32). If a super MB is split into partitions smaller than or equal to 16×16, the flag will be transmitted at the 16×16 block level. For $\text{POBMC}_{\text{H-II}}$, one additional bit is used to signal the δ value.

It is observed from Table II that $\text{Bilinear}_{\text{H-I}}$ performs similarly to $263_{\text{H-I}}$. Essentially, they both use a fixed window function and thus suffer from the same problem of having to use the MV replication mechanism to address variable block-size partitioning. As expected, adapting window functions on the fly is beneficial, and the benefits of our parametric solutions are more obvious. In the IPPP case, $\text{POBMC}_{\text{H-I}}$ has an average BD-rate saving of 2.9%, with a minimum of 0.9% and a maximum of 9.6%. Adding one more OBMC option with $\delta = 0$ ($\text{POBMC}_{\text{H-II}}$) further improves performance slightly, giving, on average, 0.1–0.6% extra gains. These results provide only a lower bound on what is achievable, because the creation of POBMC predictors currently uses motion parameters[4] optimized for BMC. It is for the same reason that the gains of POBMC (and similarly, of the other OBMC schemes) over the anchor decrease radically when the IBBB or Hierarchical B structure is in use. Note that both bi-prediction and POBMC are multi-hypothesis MCP techniques, and we are comparing a well-optimized bi-prediction with a suboptimal POBMC. Another cause of this performance decline is the increased use of the SKIP and Direct modes,

[4]Here, the motion parameters include the partition choice, the prediction type (forward-prediction, backward-prediction, or bi-prediction) for each partition, as well as the corresponding MV(s).

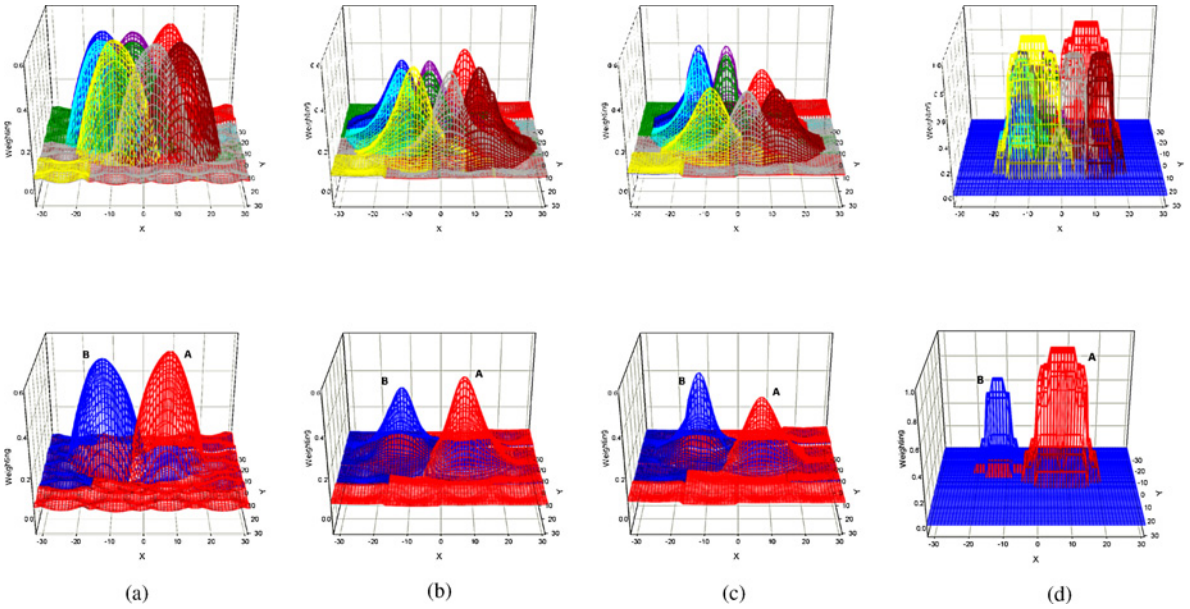


Fig. 7. Window functions overlaid on the irregular motion sampling grid shown in Fig. 6. The proposed parametric windows with a (a) non-diagonal D matrix (Clip=17, $\delta = 121$), (b) diagonal D matrix (Clip = 19, $\delta = 25$), (c) diagonal D matrix plus a MB-adaptive adjustment of δ (Clip = 19, $\delta = 16$ for 8×8 MVs and $\delta = 36$, otherwise), and (d) H.263 windows.

TABLE II

BD-RATE SAVING COMPARISON OF VARIOUS OBMC SCHEMES

Category		Non-Parametric						Parametric											
Scheme		$263_{H\text{-}I}$			$Bilinear^{*}_{H\text{-}I}$			$Tao_{H\text{-}I}$			$POBMC_{H\text{-}I}$			$POBMC_{H\text{-}II}$			$POBMC_{H\text{-}II}$+MD		
GoP		IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB
HD	S03	0.2	0.8	0.4	0.4	0.9	0.4	1.0	1.4	0.6	5.0	2.3	1.2	5.1	2.7	1.7	7.2	2.8	1.8
	S04	0.3	0.3	0.3	0.3	0.3	0.4	0.7	0.5	0.5	0.9	0.9	1.2	1.4	1.1	1.9	2.6	1.4	2.0
	S05	0.4	0.7	0.2	0.5	0.7	0.3	1.7	1.4	0.5	3.3	2.3	1.0	3.6	2.6	1.6	5.9	2.8	1.6
	S06	2.0	0.7	0.3	2.1	0.8	0.2	2.3	1.2	0.4	3.5	2.1	1.1	3.6	2.5	1.6	5.7	2.9	1.6
	S07	2.9	1.1	0.9	3.1	1.0	0.9	7.8	1.9	1.4	9.6	3.1	3.2	10.2	3.7	3.9	13.6	4.0	4.2
	Average	1.2	0.7	0.4	1.3	0.7	0.4	2.7	1.3	0.7	4.3	2.1	1.5	4.8	2.5	2.1	7.0	2.8	2.2
WVGA	S08	−0.2	−0.1	−0.2	−0.1	−0.2	0.2	1.0	0.4	0.4	1.8	1.0	0.7	1.9	1.5	1.1	3.4	1.9	1.4
	S09	0.3	0.4	0.3	0.4	0.3	0.3	1.4	0.8	0.5	2.6	1.3	1.1	2.9	1.6	1.3	4.6	1.8	1.4
	S10	0.0	0.2	−0.1	0.2	0.3	0.2	1.3	0.4	0.4	1.4	0.8	0.7	1.8	0.9	0.9	3.5	0.9	1.0
	S11	0.2	0.6	0.3	0.3	0.7	0.3	1.5	0.8	0.4	2.0	1.4	1.0	2.4	1.6	1.4	4.0	1.6	1.4
	Average	0.1	0.3	0.1	0.2	0.3	0.3	1.3	0.6	0.4	2.0	1.1	0.9	2.3	1.4	1.2	3.9	1.6	1.3
QWVGA	S12	0.1	0.3	0.2	0.1	0.4	0.2	1.5	0.5	0.4	2.2	1.1	0.6	2.4	1.6	0.8	4.0	1.8	0.9
	S13	0.8	0.8	0.4	0.9	0.7	0.4	2.1	1.4	0.6	3.3	2.3	1.2	3.8	2.5	1.2	5.9	3.0	1.2
	S14	−0.1	−0.1	−0.2	0.1	0.2	−0.1	1.0	0.4	0.3	1.4	0.9	0.7	1.7	1.3	0.9	3.0	1.5	1.1
	S15	0.1	0.2	0.2	0.1	−0.1	0.2	1.1	0.5	0.4	1.1	0.9	0.6	1.4	1.3	0.7	2.7	1.7	0.7
	Average	0.2	0.3	0.2	0.3	0.3	0.2	1.4	0.7	0.4	2.0	1.3	0.8	2.3	1.7	0.9	3.9	2.0	1.0
720p	S16	1.0	0.3	0.6	1.1	0.2	0.7	1.4	0.5	0.8	2.1	1.0	1.5	2.4	1.5	2.4	4.1	1.8	2.5
	S17	1.6	1.8	0.8	1.5	2.0	0.7	2.2	3.3	1.2	4.1	5.5	2.2	4.7	5.8	2.6	7.7	6.2	2.6
	S18	1.0	0.5	0.5	1.2	0.4	0.6	1.5	0.8	0.8	2.9	1.4	1.5	3.5	1.7	1.8	5.3	1.8	2.0
	Average	1.2	0.8	0.6	1.3	0.8	0.7	1.7	1.5	0.9	3.0	2.7	1.7	3.5	3.0	2.3	5.7	3.3	2.4
Overall		1.1	0.6	0.4	1.2	0.6	0.4	1.9	1.0	0.6	2.9	1.8	1.2	3.3	2.1	1.6	5.2	2.4	1.7

*The bilinear window function shown in Fig. 3(c) is used.
Positive values mean bitrate savings, whereas negative values mean bitrate increments.

especially in the Hierarchical B sequence. In those modes, POBMC is not functioning. Even so, $POBMC_{H\text{-}II}$ still yields 0.9–5.8% and 0.7–3.9% BD-rate savings in the IBBB and Hierarchical B sequences, respectively.

The obvious inefficiency of POBMC can be improved by optimizing motion parameters. The rightmost column ($POBMC_{H\text{-}II}$+MD) of Table II presents the results when partition choices (and only partition choices) are additionally adapted for POBMC at a higher computational cost (Appendix). As can be seen, the performance improves further in the IPPP case, adding up to an average saving of 5.2%, whereas the improvements are much smaller in the remaining cases. To see why, Fig. 8 contrasts the mode distributions before and after this optimization, where the percentage of each mode indicates its average spatial coverage in a frame, i.e., the number of pixels coded by that particular mode. It is observed that despite the use of suboptimal motion parameters, POBMC is still more efficient than single-hypothesis MCP. It thus tends to favor the use of larger partitions, thereby reducing the overhead for signaling motion parameters. However, the impact of this suboptimality can be such that the superiority of POBMC over bi-prediction becomes modest. This explains why POBMC is enabled less frequently in the IBBB or Hierarchical B sequence and why the increase in the use of larger partitions is not as significant as in the former case. Nevertheless, it is expected to

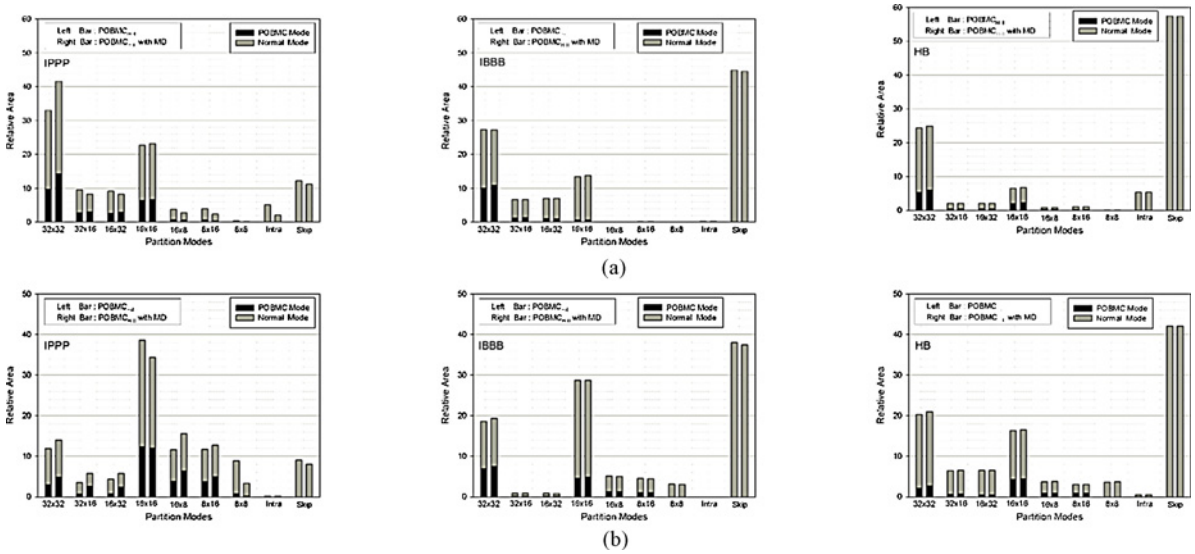


Fig. 8. Mode distribution comparison of $\text{POBMC}_{\text{H-II}}$ and $\text{POBMC}_{\text{H-II}}$+MD. (a) QP22, S03 (Kimono). (b) QP22, S13 (BQSquare).

TABLE III
COMPARISON OF IN-LOOP PREDICTION FILTERS

Filter	Function	Predictor Formulation	Filter Optimization
POBMC	Motion uncertainty	$P_k(\mathbf{s}) = \sum_{i\in\mathcal{I}_1} w_i(\mathbf{s}) I'_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))$	$E\left\{(I_k(\mathbf{s}) - P_k(\mathbf{s}))^2\right\}$
EAIF	Aliasing	$P_k(\mathbf{s}) = \sum_{i\in\mathcal{I}_2} w_i(\mathbf{v}(\mathbf{s}_c)) I'_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)+\mathbf{d}_i)$	$\sum_{\mathbf{s}}(I_k(\mathbf{s}) - P_k(\mathbf{s}))^2$
QALF	Quantization noise	$P_{k-1}(\mathbf{s}) = \sum_{i\in\mathcal{I}_3} w_i I'_{k-1}(\mathbf{s}+\mathbf{d}_i)$	$\sum_{\mathbf{s}}(I_{k-1}(\mathbf{s}) - P_{k-1}(\mathbf{s}))^2$

I'_{k-1} : reference frame with coding noise.

TABLE IV
BD-RATE SAVING COMPARISON OF VARIOUS IN-LOOP FILTERS

Scheme		$\text{POBMC}_{\text{H-II}}$+MD			EAIF			QALF		
GoP		IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB
HD	S03	7.2	2.8	1.8	9.8	5.6	2.9	12.8	9.3	9.1
	S04	2.6	1.4	2.0	2.6	2.0	2.2	2.8	3.2	4.4
	S05	5.9	2.8	1.6	3.2	2.1	2.8	5.8	4.4	5.7
	S06	5.7	2.9	1.6	7.2	4.3	3.5	8.9	6.3	6.0
	S07	13.6	4.0	4.2	9.3	8.7	7.4	10.3	15.0	11.7
	Average	7.0	2.8	2.2	6.4	4.5	3.8	8.1	7.6	7.4
WVGA	S08	3.4	1.9	1.4	6.2	8.1	3.9	10.7	11.1	7.9
	S09	4.6	1.8	1.4	4.0	3.4	2.3	4.7	3.8	3.2
	S10	3.5	0.9	1.0	4.5	2.6	3.3	5.2	7.9	9.1
	S11	4.0	1.6	1.4	1.9	1.5	1.1	2.2	2.5	2.4
	Average	3.9	1.6	1.3	4.2	3.9	2.6	5.7	6.3	5.7
QWVGA	S12	4.0	1.8	0.9	3.7	2.2	2.4	5.3	3.2	3.5
	S13	5.9	3.0	1.2	8.8	8.0	7.5	9.1	15.0	18.0
	S14	3.0	1.5	1.1	2.0	1.7	1.4	2.8	3.6	4.1
	S15	2.7	1.7	0.7	0.5	0.5	0.5	1.8	1.3	1.9
	Average	3.9	2.0	1.0	3.8	3.1	3.0	4.7	5.8	6.9
720p	S16	4.1	1.8	2.5	7.1	4.7	6.3	12.4	8.1	8.2
	S17	7.7	6.2	2.6	8.4	10.1	7.6	11.7	13.0	10.7
	S18	5.3	1.8	2.0	8.4	4.5	6.6	10.5	6.6	7.6
	Average	5.7	3.3	2.4	8.0	6.4	6.9	11.5	9.2	8.8
Overall		5.2	2.4	1.7	5.5	4.4	3.9	7.3	7.1	7.1

perform better if the MVs and prediction directions are further optimized.

2) *Comparison of In-Loop Filters and the Combined Effects:* Having compared the compression performance of various OBMC schemes, we now proceed to show how POBMC performs relative to EAIF and QALF. We begin by contrasting the functions, implementations, and optimization criteria of these filters in Table III. As can be seen, they all form a predictor (at every pixel position) from a weighted average of the reference samples, using the Wiener or least-squares criterion. However, what samples are involved depends on their main functions. For example, to address motion uncertainty, POBMC takes in those pointed to by the current and neighboring block MVs, $I'_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))$, some of which may be at sub-pixel positions. To generate sub-pixel samples, EAIF interpolates between adjacent integer samples, with a filter support lying mainly around $\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)$, where $\widehat{\mathbf{v}}(\mathbf{s}_c)$ is the current MV in integer precision. The small deviation $\mathbf{d}_i$ (in units of integer samples) from $\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)$ partly helps to mitigate the problem of motion uncertainty. In complete analogy with EAIF, QALF also linearly combines nearby integer samples, but for a purpose of smoothing out quantization noise present in the reference frame. The filtered image $P_{k-1}(\mathbf{s})$, when used for MCP, yields a form similar to that of EAIF: $P_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)) = \sum_{i\in\mathcal{I}_3} w_i I'_{k-1}(\mathbf{s} + \widehat{\mathbf{v}}(\mathbf{s}_c) + \mathbf{d}_i)$. While averaging the reference samples, these filters all reduce noise to some extent. Their functions overlap, but each has its own emphasis. As a result, they possess very different filter characteristics: POBMC has a pixel-adaptive filter function, whereas that of EAIF and that of QALF are sub-pixel-dependent and fixed,[5] respectively.

Table IV compares the BD-rate savings of $\text{POBMC}_{\text{H-II}}$+MD, EAIF, and QALF. In the IPPP case, POBMC and EAIF offer a nearly identical average BD-rate saving (∼5.3%), although the results in individual test sequences are highly variable. The gain of QALF, by contrast, is about 2% higher. In particular, they all perform better in high-resolution sequences. It is worth noting that the performance impact due to the use of bi-prediction on EAIF or QALF is minimal to moderate (see the results for the IBBB and Hierarchical B structures). The impact is mostly negative for EAIF, but is positive in quite a few sequences for QALF. Overall, both exhibit a performance decline in these prediction sequences; however, the performance drop relative to the IPPP setting is comparatively much smaller. The reasons are twofold: first, these tools have less overlap with bi-prediction than POBMC, in terms of functionalities, and second, the SKIP and Direct modes can still benefit from enabling them.

Although POBMC has the least gain among these filters, it can be combined more efficiently with either of the other two filters. Table V presents the BD-rate savings of all possible joint applications of these filters. An interesting observation is that the combination of POBMC and QALF performs very close to or better than that of EAIF and QALF, even in the IBBB or Hierarchical B case where the single use of EAIF outperforms that of POBMC. This leads us to suspect that a considerable part of the gain from EAIF is actually due to an attenuation in quantization noise. In this regard, EAIF shares similarities with QALF. This conjecture is supported by another observation that EAIF is still advantageous to high-resolution sequences, in which the signal aliasing is supposed to be less severe. The results in the rightmost column of Table V also indicates that the benefits of additionally enabling EAIF on top of POBMC+QALF are quite limited, except in few low-resolution sequences, such as S10 and S13, where the aliasing may still be significant.

[5]Note that some advanced QALF algorithms [10], [15] feature a pixel-adaptive filter.

B. Complexity Analyses

1) *Encoding and Decoding Times:* This section compares the software runtimes of the algorithms discussed above to provide a rough indication of their complexity characteristics. For the runtime measurements, a single machine equipped with Intel Core i7-860 CPU (2926 MHz), 8 GB RAM, and SATA-2 hard drive is used to run encoding or decoding in a single thread. The execution time is measured on Windows Vista 64-bit SP1 using NTimer. YUV writing is enabled during encoding. When interpreting the results, one should be aware that the software runtime can be more related to the implementation quality of an algorithm than to its intrinsic complexity.

The upper half of Table VI[6] presents the encoding times of these algorithms in the form of average time ratios[7] relative to the anchor encoding. As can be seen from the left most column, adding OBMC option increases the encoding time by 30–50%. This is attributed to the extra computation required for mode decision and pixel-adaptive weighting. Moreover, computing weighting coefficients by parametric solutions (e.g., $\text{Tao}_{\text{H-I}}$ and $\text{POBMC}_{\text{H-I}}$) involves floating-point arithmetic, thereby taking 10–20% longer encoding time, and as expected, the more flexible $\text{POBMC}_{\text{H-II}}$ increases the encoding time further (˜10%). When $\text{POBMC}_{\text{H-II}}$+MD is in use, the encoding time almost doubles due to multi-pass R-D based mode decision. Similar results can also be seen in EAIF. By contrast, QALF introduces only a moderate time increase (20–45%), the reason probably being that determining restoration filter and segmentation map may still be computationally less demanding than performing mode decision, which requires actually coding all the MBs using all possible options. We finally note that the impacts of these algorithms on encoding time appear consistent regardless of prediction structures.

While there is a wide variation between the encoding times of these algorithms, the difference between their decoding times is less significant. Still, performing OBMC leads to slow decoding times; relative to the anchor decoding (which decodes bit-streams produced by the anchor encoding), a 40–70% increase of decoding time is observed (the lower half of Table VI). The main reasons include the increased memory accesses needed to fetch multiple predictors and the extra operations required for weighting them in a pixel-adaptive

[6]$\text{Bilinear}_{\text{H-I}}$ is of comparable complexity to $263_{\text{H-I}}$ and gives similar runtime results.

[7]The average of the encoding time ratios of some specific algorithm and the anchor over all test sequences and QP settings.

TABLE V
BD-RATE SAVING COMPARISON OF VARIOUS COMBINATIONS OF IN-LOOP FILTERS

Scheme		POBMC+EAIF			POBMC+QALF			EAIF+QALF			All Enabled		
GoP		IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB
HD	S03	11.3	7.1	4.4	13.9	10.8	10.4	13.9	10.2	10.0	15.1	11.8	11.0
	S04	4.0	2.9	3.4	4.4	4.1	5.5	4.0	3.5	5.2	5.2	4.6	6.7
	S05	8.2	4.1	4.0	10.8	6.1	6.9	6.6	4.5	6.2	11.1	5.4	7.4
	S06	10.5	6.0	4.8	12.0	8.0	6.7	10.9	7.8	7.5	13.6	9.3	8.3
	S07	18.9	11.2	10.2	22.4	17.6	14.0	13.8	16.1	13.8	24.0	17.6	14.7
	Average	10.6	6.3	5.4	12.7	9.3	8.7	9.8	8.4	8.5	13.8	9.7	9.6
WVGA	S08	10.6	8.9	4.9	15.0	11.2	8.5	13.2	13.5	8.9	16.6	15.2	9.8
	S09	7.5	4.3	3.0	8.1	5.1	4.2	7.1	5.2	4.9	9.9	6.2	6.0
	S10	8.8	2.7	4.0	11.5	8.5	9.8	7.6	8.2	10.1	14.0	9.2	10.6
	S11	5.0	2.2	1.9	5.8	3.4	3.2	3.4	3.3	3.1	6.8	3.5	3.9
	Average	8.0	4.5	3.5	10.1	7.1	6.4	7.8	7.6	6.8	11.8	8.5	7.6
QWVGA	S12	6.0	3.1	2.9	8.2	4.6	4.0	6.1	3.8	4.5	8.3	4.9	4.9
	S13	15.9	9.9	8.3	20.0	17.4	18.7	13.8	15.9	19.6	24.6	17.7	20.3
	S14	5.3	2.3	2.2	7.1	4.3	4.8	3.7	3.8	4.4	7.8	4.6	4.9
	S15	3.2	1.4	0.7	4.0	2.4	2.3	2.1	1.6	2.4	4.3	2.7	2.8
	Average	7.6	4.2	3.5	9.8	7.2	7.5	6.4	6.3	7.7	11.3	7.5	8.3
720p	S16	9.6	5.3	7.8	14.5	8.9	9.7	13.5	8.5	8.8	14.9	9.1	10.6
	S17	14.9	14.7	9.6	18.6	17.7	10.5	12.8	13.7	12.3	18.8	18.1	13.9
	S18	11.1	5.8	8.1	14.0	7.2	8.8	12.0	7.7	10.0	14.8	9.4	10.9
	Average	11.9	8.6	8.5	15.7	11.3	9.7	12.8	10.0	10.4	16.2	12.2	11.8
Overall		9.4	5.7	5.0	11.9	8.6	8.0	9.0	8.0	8.2	13.1	9.3	9.2

TABLE VI
RUNTIME COMPARISON OF VARIOUS IN-LOOP FILTERS

Scheme		$263_{\text{H-I}}$	$\text{Tao}_{\text{H-I}}$	$\text{POBMC}_{\text{H-I}}$	$\text{POBMC}_{\text{H-II}}$	$\text{POBMC}_{\text{H-II}}$+MD	EAIF	QALF
Encoding	IPP	1.47	1.61	1.63	1.71	2.01	2.22	1.40
	IBB	1.37	1.50	1.57	1.62	1.97	2.00	1.45
	HB	1.32	1.42	1.48	1.60	2.00	2.12	1.25
Decoding	IPP	1.40	1.48	1.64	1.68	1.73	1.41	1.53
	IBB	1.35	1.45	1.51	1.57	1.66	1.25	1.56
	HB	1.48	1.54	1.62	1.64	1.68	1.46	1.43

manner. Another cause of this increase is an implementation expedient, which requires parsing the bit-stream twice (Appendix). Again, the parametric solutions incur a higher penalty than the non-parametric ones, and ours appear to be more complex. The latter, however, is not because our schemes are computationally more demanding, but because the OBMC mode is selected more often when they are in use. Recall that these algorithms allow the encoder to switch adaptively between BMC and OBMC. We also find that when using non-parametric windows, OBMC exhibits similar decoding complexity characteristics to EAIF or QALF. Essentially, they all perform filtering of pixel intensities based on pre-computed filters.

2) *Simplification:* In the current implementation of POBMC (and the other OBMC schemes), generating a prediction value for a pixel may require MVs associated with the neighboring blocks to its right or below. Such a non-causal access of MVs requires motion parameters for the entire picture to be first generated and buffered, which induces a large delay in both the encoding and the decoding processes. This, however, can be avoided by utilizing only those MVs in the causal neighborhood. Some performance loss is expected, as the pixels in the bottom-right quarter of a prediction block may not benefit much from OBMC. Table VII compares the BD-rate savings and the runtimes of $\text{POBMC}_{\text{H-II}}$+MD and its simplified version based on this notion. Specifically, the simplifications include a causal MV access and the use of only one δ option (δ = 16). From the results, we see that the performance of the simplified POBMC is only slightly worse than that of $\text{POBMC}_{\text{H-II}}$+MD. However, both the encoder and the decoder run much faster. This is mainly because on the encoder side, the determination of motion compensation methods and partition choices can now be done in one single pass, and on the decoder side, there is no need to first extract all motion parameters.

TABLE VII
BD-RATE AND RUNTIME COMPARISONS OF $\text{POBMC}_{\{H-II\}}$+MD AND ITS SIMPLIFIED VERSION

Scheme	$\text{POBMC}_{\text{H-II}}$+MD			Simplified		
GoP	IPP	IBB	HB	IPP	IBB	HB
HD	7.0	2.8	2.2	5.9	2.4	2.3
WVGA	3.9	1.6	1.3	3.2	1.5	1.1
QWVGA	3.9	2.0	1.0	3.3	1.9	0.9
720p	5.7	3.3	2.4	4.5	3.2	2.3
Overall	5.2	2.4	1.7	4.3	2.2	1.6
Time Ratios						
Encoding	2.01	1.97	2.00	1.46	1.40	1.33
Decoding	1.73	1.66	1.68	1.34	1.32	1.30

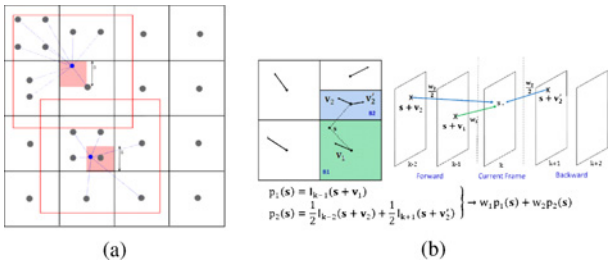


Fig. 9. (a) Sliding window mechanism used for MV selection in POBMC, where the blue dots indicate prediction pixels and the gray dots denote MV sampling points. (b) Bi-prediction extension of POBMC.

To further reduce the complexity of POBMC, additional constraints can be placed on the number of MVs involved and on their ranges. Our preliminary study shows that using only four neighboring MVs, truncated to be in the range of current MV +/− four pixels, can achieve 10–15% runtime reductions without introducing noticeable performance degradation. In addition, the techniques discussed in [18] can be applied to POBMC. Last, we want to point out that although (11) inevitably requires floating-point arithmetic, (12) can have a fixed-point implementation. This can be seen by considering the prediction of a pixel based on three MVs. In this case, the OBMC weight associated with one of these MVs is computed as

$$w_1^* = \frac{\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)}}{\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)} + \frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_2)} + \frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_3)}} = \frac{\widehat{r}^2(\mathbf{s},\mathbf{s}_2)\widehat{r}^2(\mathbf{s},\mathbf{s}_3)}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)\widehat{r}^2(\mathbf{s},\mathbf{s}_2) + \widehat{r}^2(\mathbf{s},\mathbf{s}_2)\widehat{r}^2(\mathbf{s},\mathbf{s}_3) + \widehat{r}^2(\mathbf{s},\mathbf{s}_3)\widehat{r}^2(\mathbf{s},\mathbf{s}_1)}.$$

Obviously, with proper scaling and rounding, the computations of the numerator, the denominator, and the quotient can all be done in fixed-point arithmetic, even though it still takes quite some work to obtain the result. Compared with (2), which is adopted by the parametric solution in [19], both (11) and (12) require less computation. While (12) need not perform matrix inverse, (11) only has to invert a smaller matrix of dimension $(L-1)\times(L-1)$, compared to $L\times L$ in (2).

VI. Conclusion

In this paper, we introduced a parametric solution for OBMC (termed POBMC) and studied its properties from both theoretical and empirical aspects. In the course, we found it convenient to approximate block MVs as motion samples taken at block centers; this expedient helped us to conceptualize the combination of OBMC with variable block-size motion partitioning. Because in practice it is far from adequate to describe the location of a block MV as a deterministic variable, we amended our solution to reveal its random nature. The novelty of our scheme was shown to require only the geometric relations between the prediction pixel and its nearby block centers for computing the window function. It thus lends itself to reconstructing temporal predictors from any sparsely and irregularly sampled motion data. The superiority of our scheme over similar previous works was confirmed by extensive experiments. Moreover, its performance was shown to be comparable to EAIF and QALF. Above all, it can work with either (or all) of them to yield an improvement that is nearly the sum of their separate effects.

Along with the coding tools in KTA2.4r1, part of this paper [5] was submitted, for subjective viewing tests, in response to the HEVC Call for Proposals issued jointly by MPEG and VCEG in April 2010 [2]. It was ranked 12 overall and 10 in low delay configurations among 27 proposals, in terms of the average mean opinion score [3]. The notion of POBMC has recently been extended to develop a reduced-overhead bi-prediction scheme [13], [14]. Owing to its promising results, the technique is currently being evaluated in a core experiment of JCT-VC.

We plan to continue this work in several directions: 1) to study the quantization effect of POBMC coefficients; 2) to develop efficient motion sampling patterns for POBMC; and, finally, 3) to combine POBMC with advanced decoder-side motion inference techniques.

Appendix

This appendix elaborates on the implementation details of POBMC. The intent here is to help the reader to have a better understanding of its complexity characteristics and software runtimes. In our current implementation, generating a prediction value for a pixel by POBMC may require MVs corresponding to the neighboring blocks to its right or below. Hence, the motion parameters for the entire picture have to be first generated and buffered to ease random access. Accordingly, the picture-level encoding process must undergo several sequential passes.

1) In the first pass, the encoder determines the best motion parameters for each MB based on the conventional BMC. Here, the motion parameters include the partition choice, the prediction type (forward-prediction, backward-prediction, or bi-prediction) for each partition, as well as the corresponding MV(s).
2) In the second pass, POBMC is evaluated against BMC for each non-skipped MB to see whether the R-D performance can be improved. The creation of POBMC predictors reuses motion parameters generated in the first pass, which have been buffered for a possible non-causal access.
3) When $\text{POBMC}_{\text{H-II}}$+MD is in use, the partition choice of each MB is additionally updated once in a raster scan order to optimize motion sampling for POBMC. This is accomplished by comparing the R-D costs of coding a MB with various combinations of partition choices and motion compensation methods (POBMC and BMC). While the partition choice may be varied, the corresponding prediction types and MVs are inherited directly from the first pass; this requires the first step to also buffer the motion parameters for coding modes other than the best one. In addition, the updated partition choice for the current MB will be utilized in evaluating POBMC for next MB; again, motion parameters belong-

ing to the non-causal region are set to be the same as those generated in the first pass. Once all MBs have been processed, Step 2 is repeated based on the new partitioning to complete the encoding process.

Steps 1 and 2 are followed to implement $263_{\text{H-I}}$, $\text{Bilinear}_{\text{H-I}}$, and $\text{Tao}_{\text{H-I}}$. Since the first two schemes are constrained to use a fixed window function, the MV replication mechanism (Section III-C) is employed to address the issue of variable block-size motion partitioning.

For the same reason as stated above, the motion information also has to be extracted first on the decoder side. To do so, we have the decoder parse the bit-stream twice before the decoding process begins. Although introducing a considerable delay, this implementation bottleneck can be avoided by enabling the data partitioned slices feature in H.264/AVC, with which the motion data are transported separately from the other coded data.

In addition, to predict a pixel, POBMC involves computing a weighted sum of multiple prediction values derived from MVs in its neighborhood. The neighborhood used in this paper is specified by a sliding window centered on the 8×8 block containing the prediction pixel. That is, the MVs associated with the block centers covered by this window are selected for use [Fig. 9(a)]. In cases where some of these MVs are intended for bi-prediction, we simply apply them to each pixel within the block to generate the corresponding input to POBMC [Fig. 9(b)]. Note that all pixels in this 8×8 block share the same MVs, whereas each pixel has its own weights to associate with them. The reason for not adapting MV selection on a pixel-by-pixel basis is to avoid memory access bottleneck.

Clearly, our current implementation contains many heuristic approaches. There is still plenty of room for further improvements. For example, optimizing prediction types and MVs for POBMC is expected to offer better compression performance, and storing motion parameters only for a number of MB rows or estimating the non-causal MVs based on the causal ones can reduce storage requirements.

ACKNOWLEDGMENT

The authors would like to thank C.-H. Wu for his help and support related to this paper.

REFERENCES

[1] *Video Coding for Low Bitrate Communication*, document Rec. H.263, ITU-T, Apr. 1995.

[2] *Joint Call for Proposals on Video Compression Technology*, document N11113, ISO/IEC JTC1/SC29/WG11, Jan. 2010.

[3] V. Baroncini, J.-R. Ohm, and G. J. Sullivan, *Report of Subjective Test Results of Responses to the Joint Call for Proposals (CfP) on Video Coding Technology for High Efficiency Video Coding (HEVC)*, document MPEG2010/JCTVC-A204, ISO/IEC JTC1/SC29/WG11, Apr. 2010.

[4] G. Björntegaard, *Improvements of the BD-PSNR Model*, document VCEG-AI11, ITU-T SG16, Jul. 2008.

[5] Y.-W. Chen, T.-W. Wang, C.-L. Lee, C.-H. Wu, Y.-C. Tseng, C.-H. Chan, W.-H. Peng, C.-J. Tsai, and H.-M. Hang, *Description of Video Coding Technology Proposal by NCTU*, document MPEG2010/JCTVC-A123, ISO/IEC JTC1/SC29/WG11, Apr. 2010.

[6] B.-D. Choi, J.-W. Han, and S.-J. Ko, "Irregular-grid-overlapped block motion compensation and its practical application," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 8, pp. 1221–1226, Aug. 2009.

[7] T. Chujoh, N. Wada, and G. Yasuda, *Quadtree-Based Adaptive Loop Filter*, document JVT-C181, ITU-T SG16, Jan. 2009.

[8] G. Heising, D. Marpe, H. L. Cycon, and A. P. Petukhov, "Wavelet-based very low bit rate video coding using image warping and overlapped block motion compensation," *IEE Proc. Vis. Image Signal Process.*, vol. 148, no. 2, pp. 93–101, Apr. 2001.

[9] P. Ishwar and P. Moulin, "Switched control grid interpolation for motion compensated video coding," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3. Oct. 1997, pp. 650–653.

[10] M. Karczewicz, P. Chen, R. Joshi, X. Wang, W.-J. Chien, and R. Panchal, *Video Coding Technology Proposal by Qualcomm Inc.*, document JCTVC-A121, ISO/IEC JTC1/SC29/WG11, Apr. 2010.

[11] *KTA Software* [Online]. Available: http://iphome.hhi.de/suehring/tml/download/KTA

[12] T.-Y. Kuo and C.-C. J. Kuo, "Fast overlapped block motion compensation with checkerboard block partitioning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 6, pp. 705–712, Oct. 1998.

[13] C.-L. Lee, C.-C. Chen, Y.-W. Chen, M.-H. Wu, C.-H. Wu, and W.-H. Peng, "Bi-prediction combining template and block motion compensations," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, to be published [Online]. Available: http://mapl.nctu.edu.tw/ewchen/ICIP2011PPLUS.pdf

[14] C.-L. Lee, C.-C. Chen, Y.-W. Chen, M.-H. Wu, C.-H. Wu, W.-H. Peng, and H.-M. Hang, *Bi-Prediction Combining Template and Block Motion Compensations*, document MPEG2010/JCTVC-D175, ISO/IEC JTC1/SC29/WG11, Jan. 2011.

[15] K. McCann, W.-J. Han, I.-K. Kim, J.-H. Min, E. Alshina, T. Lee, J. Chen, V. Seregin, S. Lee, Y.-M. Hong, M.-S. Cheon, and N. Shlyakhov, *Samsung's Response to the Call for Proposals on Video Compression Technology*, document JCTVCA124, ISO/IEC JTC1/SC29/WG11, Apr. 2010.

[16] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 693–699, Sep. 1994.

[17] G. J. Sullivan and R. L. Baker, "Motion compensation for video compression using control grid interpolation," in *Proc. ICASSP*, vol. 4. Apr. 1991, pp. 2713–2716.

[18] G. J. Sullivan and M. T. Orchard, "Methods of reduced-complexity overlapped block motion compensation," in *Proc. IEEE Int. Conf. Image Process.*, vol. 2. Nov. 1994, pp. 957–961.

[19] B. Tao and M. T. Orchard, "A parametric solution for optimal overlapped block motion compensation," *IEEE Trans. Image Process.*, vol. 10, no. 3, pp. 341–350, Mar. 2001.

[20] Y.-C. Tseng, C.-H. Wu, Y.-W. Chen, T.-W. Wang, and W.-H. Peng, "On the analysis and design of motion sampling structure for advanced motion-compensated prediction," in *Proc. IEEE Int. Conf. Image Process.*, vol. 1. Sep. 2010, pp. 949–952.

[21] Z. Wang, W. Wang, Y. Lu, H. Cui, and K. Tang, "Coding mode adapted overlapped block motion compensation in H.264," in *Proc. IMACS Multiconf. Computat. Eng. Syst. Applicat.*, vol. 1. Oct. 2006, pp. 1665–1668.

[22] Y. Ye and M. Karczewicz, *Enhanced Adaptive Interpolation Filter*, document T05-SG16-C-0464, ITUT SG16, Apr. 2008.

[23] W. Zheng, Y. Shishikui, M. Naemura, Y. Kanatsugu, and S. Itoh, "Analysis of space-dependent characteristics of motion-compensated frame differences based on a statistical motion distribution model," *IEEE Trans. Image Process.*, vol. 11, no. 4, pp. 377–386, Apr. 2002.

Yi-Wen Chen was born in Taichung, Taiwan, in 1979. He received the B.S. and M.S. degrees in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2001 and 2003, respectively. Currently, he is pursing the Ph.D. degree in computer science and information engineering from National Chiao Tung University.

He is a Moving Picture Expert Group (MPEG) Delegate. Since 2006, he has actively participated in ISO's MPEG digital video coding standardization process and contributed to the development of the MPEG-4 Part 10 AVC Amd.2 multiview video coding standard. He is currently devoting himself to the development of high efficiency video coding, the next-generation video coding standard. His current research interests include video/image compression, computer vision, video signal processing, content-based video indexing and retrieval, and multimedia information systems.

Wen-Hsiao Peng was born in Hsinchu, Taiwan, in 1975. He received the B.S., M.S., and Ph.D. degrees in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, in 1997, 1999, and 2005, respectively.

From 2000 to 2001, he was with the Intel Microprocessor Research Laboratory, Santa Clara, CA, where he developed the first real-time MPEG-4 fine granularity scalability codec and demonstrated its application in 3-D, peer-to-peer video conferencing. In 2005, he joined the Department of Computer Science, NCTU, where he is currently an Assistant Professor. Since 2003, he has actively participated in the International Organization for Standardization Moving Picture Expert Group (MPEG) digital video coding standardization process and contributed to the development of the MPEG-4 Part 10 AVC Amd.3 scalable video coding standard. He has published more than 30 technical papers in the field of video and signal processing. His current research interests include high-efficiency video coding, scalable video coding, video codec optimization, and platform-based architecture design for video compression.

Dr. Peng is currently a Technical Committee Member for the *Visual Signal Processing and Communications* and *Multimedia Systems and Application* tracks for the IEEE Circuits and Systems Society. He organized two special sessions on high-efficiency video coding in ICME 2010 and APSIPA ASC 2010, and was a Technical Program Co-Chair for VCIP 2011.