Optical Flow Based Co-Located Reference Frame for Video Compression

Bohan Li, *Member, IEEE*, Jingning Han, *Senior Member, IEEE*, Yaowu Xu, *Senior Member, IEEE*, and Kenneth Rose, *Fellow, IEEE*

***Abstract*— This paper proposes a novel bi-directional motion compensation framework that extracts existing motion information associated with the reference frames and interpolates an additional reference frame candidate that is co-located with the current frame. The approach generates a dense motion field by performing optical flow estimation, so as to capture complex motion between the reference frames without recourse to additional side information. The estimated optical flow is then complemented by transmission of offset motion vectors to correct for possible deviation from the linearity assumption in the interpolation. Various optimization schemes specifically tailored to the video coding framework are presented to further improve the performance. To accommodate applications where decoder complexity is a cardinal concern, a block-constrained speed-up algorithm is also proposed. Experimental results show that the main approach and optimization methods yield significant coding gains across a diverse set of video sequences. Further experiments focus on the trade-off between performance and complexity, and demonstrate that the proposed speed-up algorithm offers complexity reduction by a large factor while maintaining most of the performance gains.**

***Index Terms*— Optical flow, video coding, hierarchical structure.**

I. Introduction

MOTION compensated prediction is a key component in video compression, which exploits temporal correlation between frames [1]–[4]. Conventionally, motion compensation involves a block-based motion search, where a matching block in the reference frame is selected as prediction for the current block. The motion vector and the prediction residual are then coded and transmitted to the decoder. Many recent video codecs employ a hierarchical coding structure wherein video frames are not encoded according to their order of display, but in a pre-defined order reflecting the layered structure. In this setting, the currently encoded frame may be predicted from both past and future frames (in terms of display order), which is referred to as bi-directional prediction. Here, two motion vectors, pointing from the current block to the two reference frames are calculated by a block matching algorithm (BMA), and are sent to the decoder. The prediction is usually generated as a combination of the two reference blocks.

However, BMAs [5]–[7] largely assume that the motion of all pixels in the current block is uniform, which is only valid in the case of purely translational motion, leaving more complex motion such as rotation, zoom and perspective effects beyond their capability. The variable block size scheme [8] employed by most recent video codecs offers some mitigation of this shortcoming by enabling subdivision into smaller blocks where needed, albeit at the cost of additional overhead.

In contrast with BMAs, a dense per-pixel motion field can capture more complex motion by assigning to every pixel its own motion vector. Optical flow estimation methods are widely used in many applications to determine the dense motion field. Basic optical flow estimation was proposed in [9]. Numerous techniques were developed over the years to enhance the optical flow estimation accuracy, e.g., [10]–[14]. Recent contributions include approaches that leverage deep learning for optical flow estimation, which were shown to offer satisfactory performance [15]–[17]. Optical flow estimation has been used in a variety of application contexts, including autonomous vision systems [18], object segmentation [19], video frame rate up-conversion [20], and many more.

Motion compensated prediction in video coding can, in principle, overcome the limitations of BMA by employing a dense motion field generated by optical flow estimation. However, the enhanced prediction comes at the cost of a considerable increase in side information. One natural remedy is to reduce the overhead by further compression of the motion field information before transmission, e.g., by employing hierarchical finite element (HFE) representation [21], or the discrete cosine transform [22]. Note that these methods trade reduction in side information for some distortion of the motion field and hence degradation of the overall prediction quality. Another approach that circumvents increasing the overhead was proposed in [23], where optical flow estimation principles were used to ultimately generate a block-level motion field. While it was reported to outperform standard BMA by incorporating optical flow-like techniques, it nevertheless suffers from the inherent limitations of block-based motion.

Another important consideration is that the decoder has access to some motion information that is not fully exploited to reduce the amount of motion information that must be transmitted. This led to a series of contributions focused on

Manuscript received November 1, 2019; revised April 12, 2020 and July 2, 2020; accepted July 19, 2020. Date of current version August 17, 2020. This work was supported by Google LLC. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Giuseppe Valenzise. *(Corresponding author: Bohan Li.)*

Bohan Li was with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA. He is now with Google LLC., Mountain View, CA 94043 USA (e-mail: bohanli@google.com).

Jingning Han and Yaowu Xu are with Google LLC., Mountain View, CA 94043, USA.

Kenneth Rose is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA.

Digital Object Identifier 10.1109/TIP.2020.3014723

"decoder-side motion vector derivation" (DMVD) [24]–[26], wherein the motion vectors of the current block are derived from previously reconstructed pixels in the neighborhood, without recourse to transmitting additional side information. DMVD typically relies on template-matching motion search at the decoder in order to derive the motion vectors. An extension proposed in [27] performs separate template-matching motion searches in regions of the searching window, and linearly combines the outcomes to obtain final prediction of the current block. While improvements are reported for these template-matching methods, their weakness is due to critically relying on the spatial correlation of motion vectors, which may not be sufficiently high in many scenarios.

Another implication of the above realization that motion information available to the decoder should be exploited, involves bi-directional prediction within a hierarchical coding structure, where the current frame is predicted from reference frames that temporally precede and succeed it. This implies that the decoder already has access to some motion information relating the bi-directional references, which can be used to predict the current frame. To exploit such motion information, in [28], block-based motion estimation is performed between the *reference frames* at both the encoder and decoder, and the estimated motion is projected to the current frame. The projected motion vectors are then used to generate a motion compensated prediction for the current frame. Similarly, instead of block-based motion vectors, a dense motion field can be estimated between the reference frames, and the motion compensated prediction can be generated accordingly [29], [30]. Since the decoder performs the same estimation, there is no need for side information and hence no need to compress the dense motion field.

Motivated by similar intuition, the Bi-directional Optical (BIO) Flow approach was proposed in [31], [32]. The approach uses conventional motion estimation to calculate and transmit the bi-directional motion vectors, but then refines the motion compensation by performing optical flow based techniques on the two-sided reference frames to obtain a dense motion field. Note that the two conventional block-based motion vectors essentially provide a better initialization for the motion estimation/refinement process, but at the cost of side information as discussed earlier, since the decoder's access to motion vectors relating the bi-directional reference frames is not exploited.

It is important to note that, when estimating the per-pixel motion field between the reference frames, the above approaches must rely on a motion model (e.g., linear motion) in order to project the motion relating the reference frames onto the current frame. In many scenarios, however, the actual motion will deviate from the presumed trajectory. Therefore, the estimate obtained by interpolation under the above assumption may exhibit a local spatial offset from the source, reflecting the deviation form assumed motion model, and cause degradation in prediction quality.

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

In addition to the basic approach, we also propose various techniques that are specifically tailored to current video codecs in Section IV. Motion field initialization is first discussed, where we re-use the motion vectors available to the decoder and derive a high-quality initialization of the motion field with the proposed method. Then, to account for the workings of advanced motion reference systems in recent video codecs such as the motion vector reference scheme in AV1 [33] and the merge mode in HEVC [3], we present algorithms to incorporate our generated motion field, as well as the offset motion vectors, into the motion vector prediction schemes. Last, but not least, recognizing that optical flow estimation in video coding has somewhat different objectives than in other applications, namely, better rate-distortion performance rather than motion precision per se, we also introduce a specifically designed confidence weight estimation scheme geared to optimizing coding performance.

A major consideration in many video coding applications is complexity, and especially decoder complexity. Section V provides an analysis of the complexity of the proposed method as well as how it is impacted by various design parameters. Then, an adjustable block-constrained optical flow estimation algorithm is presented, which serves as an effective tool to control the trade-off between coding performance and complexity.

It is experimentally shown that the proposed approach achieves significant coding performance gains. Moreover, the proposed optimization techniques offer considerable reduction in complexity at limited cost in coding performance.

Some preliminary results of the basic approach and an initial proof of concept for speed optimization appeared in a conference publication [34]. The current paper subsumes the early conference paper, and not only offers a more comprehensive presentation of the approach from basic principles, but also introduces a series of novel contributions, as detailed in Section IV, most notably as relates to integration within the practical framework of state-of-the-art video coders, and a thorough complexity analysis of the proposed method coupled with speed-performance experiments. Experimental results show that these enhancements yield an increase in performance gains by a factor of over 1.3, as compared to the gains observed in the preliminary conference paper.

II. Relevant Background

In this section, the basic optical flow estimation formulation and two cost minimization algorithms are briefly introduced. Note that for the purpose of this paper, the term "optical flow" is used interchangeably with "motion field", despite some known distinctions that are inconsequential in this context.

A. Basic Formulation of Optical Flow Estimation

Optical flow estimation is usually formulated as a Lagrangian optimization, where the cost function J is given by:

$$J = J_{data} + \lambda J_{spatial}, \tag{1}$$

where the data term, J_{data}, denotes the matching cost of an optical flow solution with respect to the data (pixel intensities) it is applied to. The spatial term $J_{spatial}$ represents the spatial continuity or coherence constraint on the optical flow. λ is the Lagrangian parameter and controls the relative weight of the spatial constraint ($\lambda \geq 0$). We next summarize a basic formulation for the two terms which will be used in this paper.

Let $I(x, y, t)$ denote the pixel intensity at location (x, y) and at time instant t, and let (u, v) denote its motion, where u is the horizontal and v is the vertical component. Assuming linear motion and constant brightness of the object, given parameter $0 \leq t_d \leq 1$ and a time interval T, we have:

$$\begin{aligned} &I(x - t_d T u, y - t_d T v, t - t_d T) \\ &\quad = I(x + (1 - t_d T)u, y + (1 - t_d T)v, t + (1 - t_d)T). \end{aligned} \tag{2}$$

This equation relates to the scenario of interest where in the current frame at time t, a pixel located at (x, y) with motion vector (u, v), is predicted from the two bi-directional reference frames at time instants $t - t_d T$ and $t + (1 - t_d)T$, and specifically from pixels located at $(x - t_d T u, y - t_d T v)$ and $(x + (1 - t_d)T u, y + (1 - t_d)T v)$, respectively. Here, T represents the time interval between the two reference frames, and is hereafter normalized to $T = 1$ for simplicity of presentation.

From (2), one natural choice for J_{data} is the sum of squared pixel differences between the two reference frames after motion compensation:

$$J_{data} = \sum \{I_{-t_d} - I_{(1-t_d)}\}^2, \tag{3}$$

where $I_{-t_d} = I(x - t_d u, y - t_d v, t - t_d)$ and $I_{(1-t_d)} = I(x + (1 - t_d)u, y + (1 - t_d)v, t + (1 - t_d))$, corresponding to the pixel values in (2) with T normalized to 1, and the summation is over every pixel in the current frame.

In order to obtain a data term that yields a simpler relationship with respect to u and v, as introduced in [9], we apply first-order Taylor expansion to I_{-t_d} and $I_{(1-t_d)}$ about (x, y, t) and substitute into (3). After reordering terms, we arrive at the following equation:

$$J_{data} = \sum (I_x u + I_y v + I_t)^2. \tag{4}$$

Here I_x, I_y and I_t denote the partial derivatives with respect to x, y and t, respectively.

To formulate the spatial term, consider the simple 4-directional 2D Laplacian filter:

$$\Delta u_{x,y} = -4u_{x,y} + u_{x-1,y} + u_{x+1,y} + u_{x,y-1} + u_{x,y+1}, \tag{5}$$

and define the spatial term as:

$$J_{spatial} = \sum \{(\Delta u)^2 + (\Delta v)^2\}. \tag{6}$$

It is worth noting that there exist optical flow estimation schemes that utilize more complicated cost functions than (4) and (6), with possibly better accuracy. However, the focus of this paper is on how to utilize optical flow estimation within video coding applications, as well as on how to optimize the overall scheme for the video coding scenario, rather than the specific implementation of the optical flow estimation itself. The formulation described in this section was selected due to its simplicity, which better satisfies the practical constraints of video coding applications. We note that more sophisticated optical flow estimation techniques can be similarly incorporated into the proposed scheme.

B. Cost Minimization Algorithms

Given N pixels of interest, we use a compact vector notation for their horizontal and vertical motion components: $\mathbf{x} = (u_0, u_1, \ldots, u_{N-1}, v_0, v_1, \ldots, v_{N-1})^T$. Then, noting its quadratic form, (4) can be written as:

$$J_{data} = \mathbf{x}^T D^T D\mathbf{x} - 2\mathbf{b}_{data}^T \mathbf{x} + c_{data}, \tag{7}$$

where D, $\mathbf{b}_{data}$ and c_{data} are easily obtained from (4). Specifically, D is a diagonal matrix with I_x and I_y of all pixels appearing on its diagonal, etc.

Similarly, the spatial term (6) can be written as:

$$J_{spatial} = \mathbf{x}^T L^T L\mathbf{x} - 2\mathbf{b}_{spat}^T \mathbf{x} + c_{spat}, \tag{8}$$

where L, $\mathbf{b}_{spat}$ and c_{spat} are derived from the Laplacian filter mask of (5). As will be of interest later in Section V, note that L as derived from (5) is a sparse matrix whose diagonal elements are all -4, and whose each row has additional four non-zero elements taking the value 1. Also note that (5) only defines the Laplacian filter for a general location with all neighbors available, hence a modification is needed for boundary pixels, and different strategies for this will yield different $\mathbf{b}_{spat}$ and c_{spat}. Furthermore, more complex Laplacian masks can be used in lieu of (5) (e.g., in Section IV-D, we propose an adaptive Laplacian mask). While the design choices may yield different values for L, $\mathbf{b}_{spat}$ and c_{spat}, the general derivation in this section remains valid.

It follows from (7) and (8) that the overall cost function can be written as:

$$J = \mathbf{x}^T A\mathbf{x} - 2\mathbf{b}^T \mathbf{x} + c, \tag{9}$$

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

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

$$A\mathbf{x} = \mathbf{b}. \tag{10}$$

A natural technique to solve the linear equations of (10), given the symmetric positive semi-definite property of A, is the conjugate gradient (CG) method [35]. Other iterative approaches target direct optimization of the cost function. For example, it was observed in [9] that the data term only involves the motion of the current pixel while the spatial term introduces inter-pixel cross terms, which suggests that the spatial term of a current pixel can be estimated using motion vectors from the last iteration, so as to circumvent the cross terms.

From (5), we have $\Delta u = w_c(\bar{u} - u)$ and $\Delta v = w_c(\bar{v} - v)$, where $\bar{u}$ is the average of the horizontal components of the neighboring motion vectors (and similarly for $\bar{v}$), and w_c is the center weight of the Laplacian filter mask (for example, in (5), $w_c = 4$). For iteration $k+1$, this approach uses the average from the last iteration $\bar{u}^{(k)}$ to approximate $\bar{u}^{(k+1)}$. Since $\bar{u}^{(k)}$ is considered constant during iteration $k+1$, this allows separate per-pixel optimization. The update at iteration $k+1$ for a given pixel is:

$$u^{(k+1)} = \bar{u}^{(k)} - \frac{I_x(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t)}{(w_c^2\lambda + I_x^2 + I_y^2)};$$
$$v^{(k+1)} = \bar{v}^{(k)} - \frac{I_y(I_x\bar{u}^{(k)} + I_y\bar{v}^{(k)} + I_t)}{(w_c^2\lambda + I_x^2 + I_y^2)}. \quad (11)$$

Note that this iterative approach is similar to the Jacobi iterative method for solving linear equations, and is essentially a stationary iterative method, which may suffer from slower convergence when compared to Krylov subspace methods such as CG. However, with more complicated (non-linear) cost functions, such iterative approaches may offer a more straightforward yet effective form.

In this paper, we employ the CG method as the cost minimization method, and our focus is on cost functions that are mapped to solving linear equations. Complexity analysis of the approach is provided in Section V.

III. Proposed Scheme and the Co-Located Reference Frame

A. Overview of the Basic Scheme

Consider the scenario of hierarchical frame coding structure, and specifically when the processing frame f_n, located at time n, occurs while there exist bi-directional reference frames f_{n_0} and f_{n_1} ($n_0 < n < n_1$) that have already been reconstructed (denoted as $\hat{f}_{n_0}$ and $\hat{f}_{n_1}$). Note that such reconstructed references are available to both encoder and decoder. As mentioned earlier, there is also some motion information relating these reference frames which is available to the decoder but is not fully utilized by conventional bi-directional motion compensation schemes.

To account for such motion information, as represented by the dash line in Figure 1(a), it is natural to assume linear motion between the reference frames and then perform optical flow estimation to obtain the motion field. In addition to the basic cost minimization method introduced in Section II, various techniques are utilized to improve the robustness of the algorithm and will be presented in Section III-B. Moreover, design optimization tailored for the video coding application is introduced in Section IV.

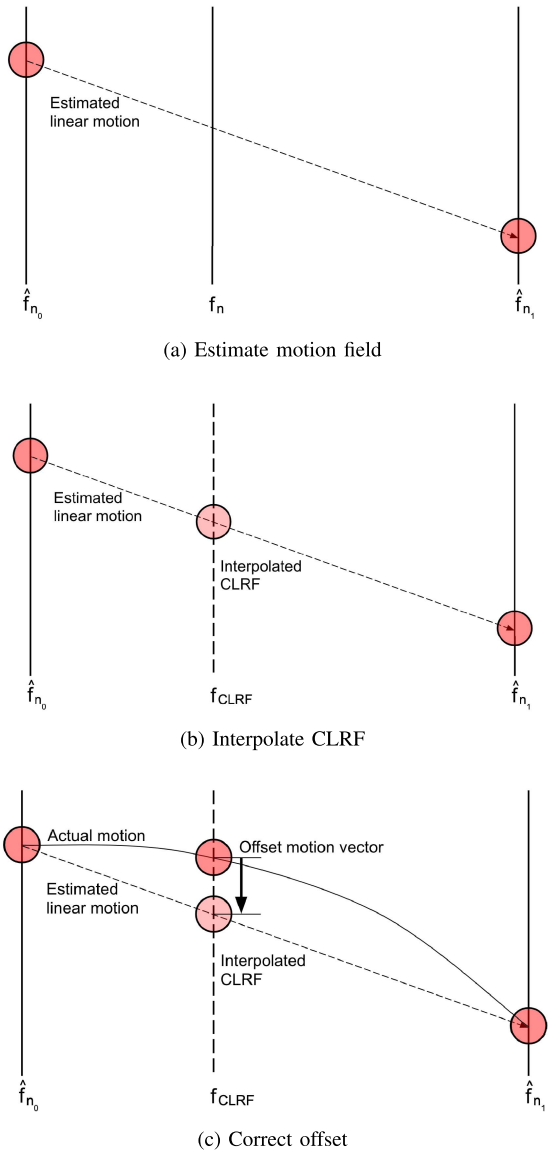


Fig. 1. Illustration of the proposed prediction scheme. First a linear motion model is assumed and the motion field is estimated. Then a co-located reference frame (CLRF) is interpolated accordingly. An offset motion vector is then calculated and sent per block to correct possible offsets from the linear assumption.

Given the estimated motion field at frame f_n, we interpolate a new frame that combines information provided by the two reference frames on frame f_n. As long as the motion field is estimated accurately, and the linear motion assumption is valid (i.e., objects move at constant velocity), the interpolated frame should be exactly co-located with f_n. Therefore, we refer to the interpolated frame as the "co-located reference frame" (CLRF), as shown in Figure 1(b). CLRF extracts the motion information between $\hat{f}_{n_0}$ and $\hat{f}_{n_1}$, and is capable of capturing complicated non-translational motion thanks to the estimated per-pixel motion field. It is important to emphasize that generating the CLRF at the decoder does not require any extra side information.

Algorithm 1 Overall Algorithm of the Proposed Bi-Directional Prediction Scheme

for every frame f_n **do**
  Determine the reconstructed reference frames $\hat{f}_{n0}$, $\hat{f}_{n1}$
  if $\hat{f}_{n0}$ or $\hat{f}_{n1}$ does not exist **then**
    Continue to next frame
  end if
  Calculate optical flow between $\hat{f}_{n0}$ and $\hat{f}_{n1}$
  Interpolate the CLRF f_{CLRF}
  for every block b in f_n **do**
    Get $\mathbf{mv}_{offset}$ associated with b
    Find block b' in f_{CLRF} given by $\mathbf{mv}_{offset}$
    Set prediction $\tilde{b} \leftarrow b'$
  end for
end for

While CLRF may be used directly as the prediction for f_n, it is important to keep in mind that the linear motion assumption is not often perfectly valid, which results in an offset between the interpolated frame and the ground truth. This offset, even if quite small, can significantly compromise the prediction quality.

The remedy, as illustrated in Figure 1(c), is to calculate an *offset motion vector per coding block*, in accordance with the block-based scheme commonly used in modern video codecs. Our experiments show this to be effective in correcting possible offsets. In this manner, CLRF is treated as a regular reference frame and the offset motion vectors are treated as regular motion vectors associated with CLRF for integration with a block-based video codec. Integration problems and solutions will be extensively discussed in Section IV-A.

It should be noted that one could consider more sophisticated motion models than the linear motion that is initially assumed in our implementation. It is nevertheless still highly unlikely to perfectly capture the actual motion, and hence offset motion vectors may still be needed. To adapt the proposed method to such models, one simply modifies the way CLRF is generated following the expected trajectories, while the rest of the algorithm remains the same.

The overall bi-directional prediction algorithm that leverages the CLRF concept, is summarized in Algorithm 1.

B. Optical Flow Estimation and CLRF Interpolation

The optical flow estimation accuracy plays a crucial part in the overall prediction quality. Note that the cost optimization method introduced in Section II depends on the assumption that the brightness of an object remains constant, and that the spatial derivatives I_x and I_y are stationary. In practice, however, the scene may be complex and such assumptions may only hold locally, potentially resulting in poor optical flow estimation accuracy.

A commonly utilized technique that introduces a *pyramid structure* is helpful in such scenarios. Here, the reference frames are resized to different resolution scales to form a pyramid, and a lower resolution optical flow is first calculated to initialize the optical flow estimation of the next level (at higher resolution). In this way, it is easier to capture large motion at lower resolution, while at higher resolution the details are further refined.

The observations that resizing to low resolution entails loss of detail and, further, that calculating the derivatives at low resolution runs the risk that the derivative filter mask extends beyond the stationary local areas, motivate the choice of a slightly different approach for the pyramid structure. First, we calculate the derivatives at the original resolution, and then resize the derivatives to the desired scale.

In addition, at each pyramid level, multiple *warping steps* are performed to improve the estimation accuracy. After calculating the optical flow at a certain step, we warp the reference frames accordingly towards the current frame f_n. Then, at the next step, the motion field is updated by estimating the optical flow between the warped reference frames, and will be used for future warping steps.

To interpolate the CLRF, we warp the two reference frames according to the final optical flow estimate, and then employ a weighted average to combine them:

$$I_{CLRF}(x, y) = (1 - t_d) I_{n_0}(x_{n_0}, y_{n_0}) + t_d I_{n_1}(x_{n_1}, y_{n_1}), \quad (12)$$

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

IV. CLRF Integration Within the Video Codec

In this section, we first introduce the video codec integration of the proposed scheme, followed by design considerations to improve the overall video coding performance.

A. Integration With the Video Codec Structure

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

At the encoder, for every frame, first determine if the CLRF option is available, i.e., there exist two reference frames in the reconstructed frame buffer, such that the current frame is in between them. Otherwise, the regular coding scheme is used. When the CLRF option is available, interpolate to generate a CLRF and use it as a candidate for single reference frame motion compensation.

In the bitstream, to signal the reference frame used for a certain inter block, a flag is first sent (if CLRF is available for the current frame) to signal whether the block is using CLRF as the reference frame. If using CLRF, the regular coding scheme is used with CLRF in terms of the associated motion vector, residuals, etc. If CLRF is not used for the block, the regular bitstream syntax for other reference frame candidates will then be coded.

The decoder can also infer the availability of CLRF for every frame. When the CLRF option is available, the decoder also calculates the optical flow, interpolates to generate a CLRF and, for every block, reads the relevant flag to determine whether CLRF is used.

Our proposed scheme uses CLRF as an additional reference frame. As with regular reference frames, block based motion search is performed on CLRF, and the selected offset motion vectors are treated as regular motion vectors.

We implemented the proposed approach within the AV1 codec, noting that integration with other block-based video codecs is possible and straightforward.

B. Motion Field Initialization

The optical flow estimation, as also stated in Section III-B, relies on constant object brightness across frames, as well as stationary spatial derivatives. In practice this condition only holds (approximately) in local areas over short time intervals. Therefore, when the motion vector for an object is large, relative to the local area, it compromises the optical flow estimate. This scenario occurs frequently in dynamic video sequences. Moreover, when the two reference frames are far apart in the hierarchical coding structure, the resulting motion between them is often quite large.

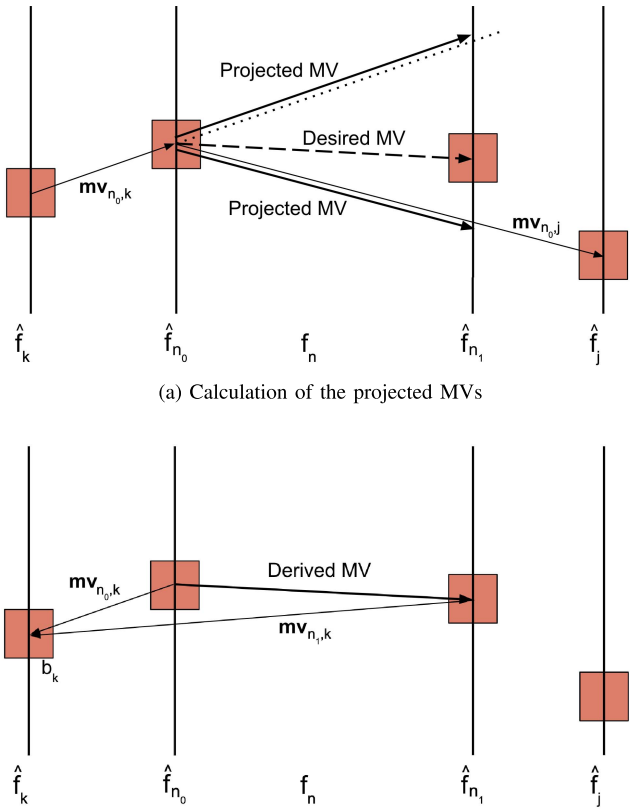
The above problem can be mitigated by better initialization of the motion field, where the regions pointed to by the initial motion vectors already belong to the same local area and the intensity constancy condition is satisfied. Thus, performing optical flow estimation on top of such initialization will significantly enhance the accuracy.

In many optical flow estimation applications, motion search is first performed between the frames to provide a good initialization. However, this may not be the best option for our proposed method considering the facts: 1) motion search at the decoder adds considerable complexity which is not desirable in many video coding applications; 2) unlike the classical optical flow estimation setting where only the two reference frames are available, in video coding the two reference frames have already been analyzed by the encoder and useful information may have already been extracted and buffered for the current frame.

Therefore, we propose the following initialization method, which does not require extensive motion search. The key idea is to utilize the motion vectors associated with the reference frames, which are already available at both the encoder and decoder.

Consider the scenario with the current frame f_n, a preceding reference frame $\hat{f}_{n_0}$ and a subsequent reference frame $\hat{f}_{n_1}$. First, since the initialization essentially tries to coarsely estimate the motion between $\hat{f}_{n_0}$ and $\hat{f}_{n_1}$, we look at every inter-predicted block in the reference frames, and if its motion vector points to the other reference frame, then this motion vector is considered as part of the initialization of the motion field. We refer to such motion vectors that relate one reference frame to the other as the *direct MVs*.

While the direct MVs offer a high quality initialization for the motion field, there may be many blocks in the reference frames whose MVs do not point to the other reference frame, resulting in many uninitialized regions in the current frame.



(a) Calculation of the projected MVs

(b) Calculation of the proposed derived MVs

Fig. 2. Examples of different methods to calculate the motion field initialization. For a certain block in $\hat{f}_{n_0}$, if its associated motion vector does not point to $\hat{f}_{n_1}$, but other frames $\hat{f}_k$ or $\hat{f}_j$, we can calculate the projected MVs or the proposed derived MVs, and use them as initialization. Note how the proposed derived MVs in Figure 2(b) serve as a better initialization when the motion is non-linear.

In preliminary work [34], we proposed a method based on linear projection (developed from a motion vector reference scheme proposed in [33]) to reduce the number of uninitialized regions. As shown in Figure 2(a), taking $\hat{f}_{n_0}$ as an example, if the motion vector of a certain block in $\hat{f}_{n_0}$ does not point to $\hat{f}_{n_1}$, we project this motion vector to $\hat{f}_{n_1}$ by assuming linear motion. Such motion vectors are referred to as *projected MVs*.

However, although the projected MVs may fill in the uninitialized regions, their quality is questionable as they depend heavily on the motion linearity assumption (illustrated in Figure 2(a) by projected MVs that differ significantly with the desired initialization).

Recognizing this problem, this paper proposes another approach to provide more reliable alternative initializations in addition to the direct MVs. Figure 2(b) illustrates how this approach works. Still taking $\hat{f}_{n_0}$ as an example, if the motion vector of a certain block, $\mathbf{mv}_{n_0,k}$, is not pointing to $\hat{f}_{n_1}$, but to block b_k in frame $\hat{f}_k$, we check whether there exists any motion vector $\mathbf{mv}_{n_1,k}$ that lies between $\hat{f}_{n_1}$ and $\hat{f}_k$, pointing to/from b_k. If such a motion vector exists, then the difference (if the two MVs point to the same frame $\hat{f}_k$) or the sum

(if $\mathbf{mv}_{n_1,k}$ points from $\hat{f}_k$ to $\hat{f}_{n_1}$) of the two motion vectors represents the motion between $\hat{f}_{n_0}$ and $\hat{f}_{n_1}$, and thus can be used as initialization. We refer to these motion vectors as the *derived MVs*. As can be seen from Figure 2(b), the derived MVs do not rely on the linear motion assumption, and hence are more reliable than the projected MVs.

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

C. Motion Vector Prediction With CLRF

In Section IV-A, we proposed to integrate the CLRF into video codecs as a regular reference frame, and also regard its associated offset motion vectors as regular motion vectors. Note, however, that the offset motion vectors, though treated as regular, do not represent actual motion of the block, but rather its deviation from the assumed linear motion model.

In this section, we first point out that the difference in physical meaning between offset motion vectors and regular motion vectors could undermine the motion vector prediction scheme used by video codecs. We thus provide a novel approach to perform motion vector prediction in conjunction with CLRF, to overcome these difficulties.

In video coding, the motion vector of every inter-predicted block must be transmitted to the decoder. To improve the coding efficiency, motion vectors are first predicted and the prediction residuals are then coded. In many video codecs, motion vectors of spatially and temporally neighboring blocks are used for such prediction. The motion field initialization introduced in Section IV-B can also serve as a temporal prediction of motion vectors [33].

However, in the case of CLRFs, the motion vector we predict from could be an offset motion vector associated with a CLRF, while the motion vector to be predicted is a regular motion vector. As explained, these two motion vectors differ in their physical interpretation, which could result in low-quality prediction, potentially breaking the motion vector prediction loop.

The solution to this problem is based on the observation that although the offset motion vector does not represent the actual motion, together with the estimated optical flow, the actual motion can be derived for that neighboring block. As shown in Figure 3, which takes the spatial motion vector prediction as an example (assuming the CLRF and the current block share the same reference frames), the derivation is done by the following steps:

First, find the adjusted location given by the location of the neighboring CLRF block and its offset motion vector $\mathbf{mv}_{offset}$. At the adjusted location, find its motion field given by the optical flow estimation. By averaging the motion field in the adjusted region, form the motion vectors pointing from this adjusted location to f_{n_0} and f_{n_1}, denoted as $\mathbf{mv}_{mf,0}$ and $\mathbf{mv}_{mf,1}$. Finally, the motion vector predictions are given by

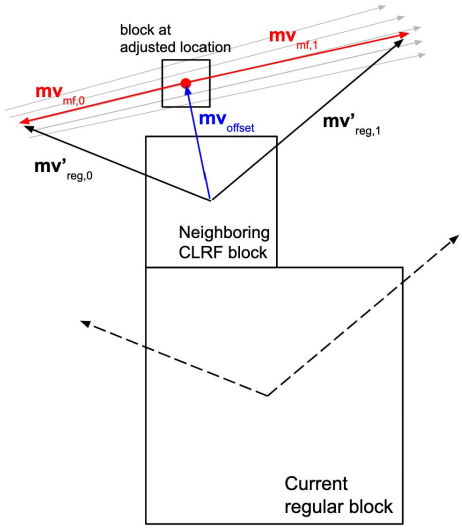


Fig. 3. Predicting regular motion vectors from an offset motion vector.

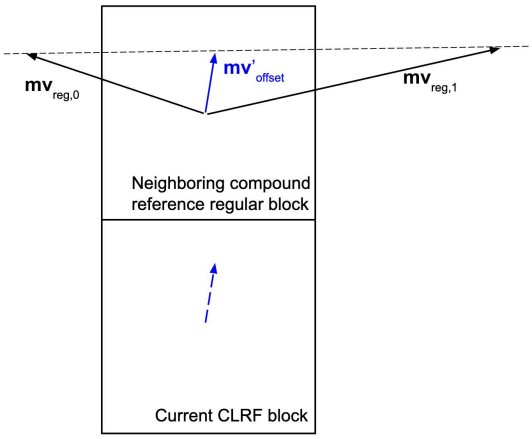


Fig. 4. Predicting an offset motion vector from regular motion vectors.

vector addition, i.e., adding $\mathbf{mv}_{offset}$ to $\mathbf{mv}_{mf,0}$, or $\mathbf{mv}_{mf,1}$, respectively. The resulting predictions are depicted by the dashed arrows in the figure.

Similarly, there are also situations where we try to predict an offset motion vector from regular vectors. For such situations, we apply the same reasoning, but in a "reversed" manner. As illustrated in Figure 4, the motion offset prediction is derived by calculating the linearly weighted average of the regular motion vectors $\mathbf{mv}_{reg,0}$ and $\mathbf{mv}_{reg,1}$:

$$\mathbf{mv}'_{offset} = (1 - t_d)\mathbf{mv}_{reg,0} + t_d\mathbf{mv}_{reg,1}. \tag{13}$$

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

D. Confidence Based Optical Flow Estimation

Optical flow estimation relies heavily on the various assumptions on the stationarity properties of the signal, and there exist various scenarios when it will not provide a good estimate. Such scenarios typically occur when some

information in one reference frame is not available in the other (e.g., due to occlusion, appearance of new objects, etc.), and typically affect a portion of the whole frame. Such affected areas are identified as low-confidence areas, while areas where we can accurately estimate the optical flow are high-confidence areas.

Note that the cost functions J_{data} and J_{spat} in (4) and (6) are summations of costs across the entire frame, and generally combine both high-confidence and low-confidence areas. This might compromise the accuracy in the high-confidence area in order to satisfy spatial constraints and to provide better accuracy on the average. Similarly, in many optical flow based applications, compromises in terms of actual distortion from the ground truth (e.g., in the MSE sense) are made for the interpolated frame given by the optical flow estimate, in order to avoid obvious visual artifacts.

However, we emphasize that this is not the case for our scheme. First of all, CLRF is not a display frame, therefore visual artifacts are not a concern, whereas the actual prediction quality with respect to the ground truth plays a more prominent role. Furthermore, CLRF is not used as in its entirety to predict the current frame, but rather as a collection of candidate reference blocks for predicting a given coding block. Therefore, the encoder is more likely to select CLRF as reference in the high-confidence areas, than the low-confidence areas where the encoder will often choose from other, conventional prediction references.

We thus conclude that, unlike other classical applications of optical flow estimation, our scheme should place more emphasis on the quality of the high-confidence areas, which is more likely to contribute to the overall coding performance. In other words, in low-confidence areas, it is acceptable to sacrifice the quality of the optical flow estimate (since they are unlikely to be used for prediction), so as to avoid compromise of the high-confidence areas.

To implement the above principle, the confidence level of each pixel is first estimated. There have been various approaches to estimate the confidence of optical flow estimation [12], [13]. To avoid the complexity of such approaches, a simple approach is utilized in our proposed method. Based on the pixel intensity constancy constraint, if the two pixels at the two reference frames, matched by a calculated motion vector, exhibit very different pixel intensities, then this motion vector is not to be trusted. Specifically, the confidence weight of location (x, y), denoted as $w(x, y)$, is calculated by:

$$w(x, y) = e^{-\alpha_c \sum_k \{I_0(x_0^{(k)}, y_0^{(k)}) - I_1(x_1^{(k)}, y_1^{(k)})\}^2}, \qquad (14)$$

where α_c is a parameter controlling the decay in confidence weight. Note the pixel difference is first averaged in a local area (e.g., 5×5 neighborhood of the current pixel), and then used to determine the weight, where k enumerates pixels in this local area.

If for some pixel location, its associated estimated motion vector points out of bound of the reference frame, then its confidence is defined at a very low level (0.01 in our experiments).

Given the confidence of each pixel, we then modify the Laplacian filter mask to attenuate the influence from a low-confidence pixel to its neighboring high-confidence pixels. To achieve this, for every pixel location, we look at its neighboring pixels used by the Laplacian filter mask (which, in the case of the 4-directional filter, are the horizontal and vertical neighbors). If the confidence level of the neighboring pixel is either 1) greater than the confidence of the current pixel, or 2) greater than a threshold T_w, we decide this neighboring pixel as a high-confidence pixel and set its mask value for the current pixel as 1. If the neighboring pixel is not a high-confidence pixel, we use its associated confidence weight as its mask value. The center of the mask is decided by the negative of the sum of all mask values for the neighboring pixels. In our experiments the parameters are chosen empirically ($\alpha_c = 0.001$ and $T_w = 0.2$). It is worth noting that the overall performance is not sensitive to small changes to the parameters. Refer to Figure 5 as an example.

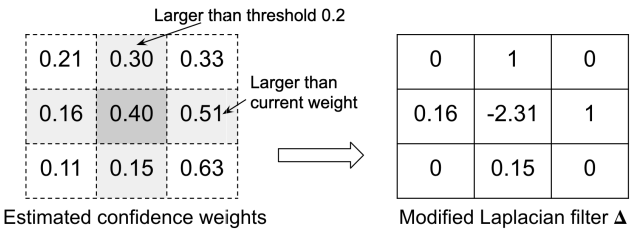


Fig. 5. Modification of Laplacian filter mask according to the confidence weight. As we are processing the current pixel (denoted by the gray block, with confidence weight 0.4), its neighbors used by the 4-directional Laplacian filter (denoted by the light gray blocks) are classified as low-confidence or high-confidence pixels, and the associated Laplacian filter mask value is modified accordingly.

The above change to the Laplacian filter effectively modifies the spatial term, and stops the propagation of estimation errors in the motion field from the low-confidence area to the high-confidence area. Additionally, the absolute value of the center mask value is also smaller when the pixel belongs to a low-confidence neighborhood, reducing the importance of that pixel in the total cost.

Similar to this center mask value, for each pixel, we also apply a weight to the data term. The weight is simply chosen as the confidence weight of the current pixel. In this way, if we are not certain about some areas, the cost of having errors in those areas is considered not as important as the high-confidence areas. This shifts the optimization focus towards the high-confidence areas, where it is more likely for the encoder to choose CLRF as the reference frame.

V. Complexity Analysis and Speed Optimization

A. Complexity Analysis of Optical Flow Estimation

From (10), given N pixels, $\mathbf{x}$ is a vector containing $2N$ variables. Therefore, minimizing the cost J in (9) involves solving $2N$ linear equations. Directly solving the inverse of a $2N \times 2N$ matrix A would incur $O(N^3)$ complexity with simple approaches such as the Gaussian elimination method.

However, since the accuracy requirement of our calculation is fairly limited, iterative solutions such as the conjugate gradient (CG) method offer computational efficiency. Generally, the complexity of CG is $O(N^2 M)$, where M is the number

of iterations and can be chosen much smaller than $2N$ for our precision requirements.

Furthermore, note that A is a sparse matrix with $O(N)$ number of non-zero elements. This is mainly due to the Laplacian filter mask, where only a fixed number of neighbors (defined by the mask) can affect the variables (u, v) associated with the current pixel. Formal proof ascertaining the degree of sparsity of A is not included here, but is straightforward to obtain by explicit consideration of $D^T D$ and $L^T L$ in (7) and (8).

Therefore, considering that the dominant complexity cost of each CG iteration lies in calculating the matrix vector product, which is $O(N)$ due to the sparsity of A, we conclude that the total complexity of optimizing J is $O(NM)$.

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

In addition, as discussed in Section III-B, the number of pyramid levels num_P and the number of warping steps num_W at each pyramid level are also factors influencing the overall complexity. Other components (e.g., median filtering of the motion field, the choice of interpolation filters, etc.) also affect the total complexity, but the main focus here is on the dominant complexity cost of solving the linear equations.

B. Speed Optimization With the Block-Constrained Algorithm

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

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

At the encoder, before encoding the current frame, we calculate the optical flow for each such block, and combine the optical flow of these blocks to form a frame-level motion field. Then, CLRF interpolation and subsequent encoding of the current frame follow the prescription of Section III.

At the decoder, however, the optical flow estimation is not done before decoding the current frame. Instead, as we decode and reconstruct a coding block, only if it uses CLRF as reference frame for motion compensation, will we perform optical flow estimation for the $h \times w$ blocks in f_{CLRF} that are needed by the current coding block. These $h \times w$ blocks are determined by the location of the current block, the offset motion vector, and the length of sub-pixel interpolation filter L (as illustrated in Figure 6). If a certain $h \times w$ block in the CLRF is already interpolated for a previously decoded coding block, then skip its calculation and use the previously interpolated result.

The above block-constrained algorithm considerably improves the speed of the proposed scheme because of the following reasons. First, the complexity of performing optical flow estimation for a block is $O(N_b M_b)$, where N_b is the number of pixels in a block ($N_b = hw$), and M_b is the number of iterations needed. Therefore the total complexity of processing all blocks in the frame is $O(\sum\{N_b M_b\}) = O(\sum\{N_b\} M_b) = O(N M_b)$. As mentioned, since N_b is much smaller than N, the number of iterations (M_b) needed to converge to a certain precision is also much smaller than M, thus reducing the total complexity by a large factor. Second, at the decoder, only a portion of $h \times w$ blocks will need optical flow estimation, since there may be many other coding blocks predicting from other references rather than from CLRF. The decoder complexity is further reduced in this way. Lastly, the optical flow estimation of each block should not interfere with each other, which ensures that highly effective parallelization can be exploited in the hardware design.

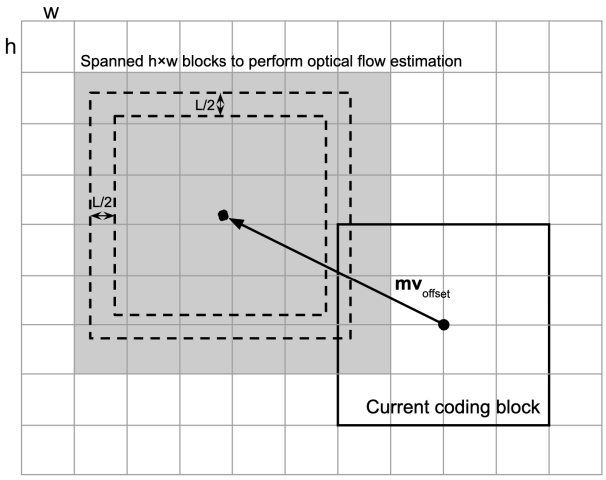


Fig. 6. Illustration of the block-based algorithm at the decoder. For the current coding block, the $h \times w$ blocks that require optical flow estimation are marked by the gray blocks, which are determined by the offset motion vector $\mathbf{mv}_{offset}$ and the length of sub-pixel interpolation filter L.

The key aspect of the block-based algorithm lies in the fact that the optical flow estimation of each block does not rely on other blocks. This enables the decoder to selectively skip certain blocks according to the usage of CLRF. To ensure such independence, the estimated optical flow of neighboring blocks should not affect the current block through the spatial constraint term. Therefore, we treat the initialization of the neighboring blocks' motion field as their actual motion. Since the initialization is already available to the decoder before the optical flow estimation, it is considered as part of the constant term c_{spat} in (8), effectively relaxing the spatial constraint across the block boundary compared to the frame-based algorithm. Moreover, noting that the initialization is not accurate, we apply a lower confidence level to the initialization to decrease its influence on the current block.

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

[1] In fact, it is proven that when $M = 2N$, CG is guaranteed to converge to the exact solution assuming no precision loss (rounding error) [35]. However in our application, such high precision is not required and we still choose $M \ll N$.

It should be noted that the block-constrained algorithm depends more heavily on the quality of motion field initialization. This is because, on the one hand, the initialization is used to handle the block boundaries, and on the other hand, the block-constrained algorithm, by its nature, works only locally, thus requiring the initialization to match the current block to the same local area for better accuracy.

The choice of the block size controls the trade-off between speed and performance, where a larger block size incurs higher complexity and yields higher quality. The below experiments used parameter values $h = w = 16$.

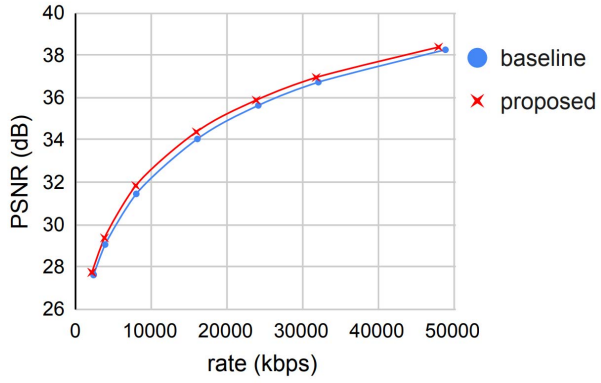
VI. Experimental Results

As mentioned in Section IV-A, the proposed scheme is generally applicable to any video codec that supports block based bi-directional motion compensated prediction to exploit temporal correlations. To evaluate its effectiveness and substantiate the benefits it offers, the proposed scheme, along with its various enhancements and optimizations, were integrated within the AV1 framework, according to the prescriptions of Section IV-A, and is compared to the baseline (libaom hashtag 3a1bd78). The choice to implement within the AV1 codec was simply because it was the most recently finalized major codec at the time this work was carried out. It must be noted that the benefits observed are strictly due to better bi-directional prediction, and are largely orthogonal to other modules of the codec, and as such are expected to be realized in conjunction with any modern video codec.

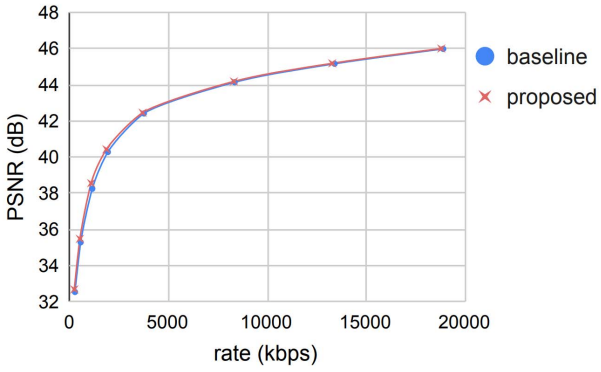
Various video sequences were tested, with different resolutions including low-res (240p, CIF), mid-res (480p, 4CIF) and hd-res (720p, 1080p). For each video sequence 150 frames are encoded at various target bit-rates (separately determined for each sequence, resulting in overall PSNR ranging between 30 and 50 dB). Hierarchical structure is enabled and the maximum length of a group of pictures is 16. In this section, we will discuss experimental results from two perspectives: the peak performance and the trade-off between complexity and performance.

For peak performance, frame-based optical flow is utilized with a three-level pyramid structure ($num_P = 3$). The number of warping steps is also set to 3 ($num_W = 3$).

The BD-rate [36] reduction compared to the baseline AV1 encoder, in terms of overall PSNR, is shown in Table I. It is evident that substantial coding gains are obtained and that the coding gains are consistent across the extensive set of testing video clips with various resolutions. The overall bit-rate reduction is 3%, 3.8% and 3.5% for low-res, mid-res and hd-res, respectively. Especially, note the relatively larger gains (approximately 8%-10%) for sequences with complex sets of moving subjects (such as crowd_run, rush_field_cuts, ice, etc.) and sequences with non-translational motions (such as station2, blue_sky, city, etc.). This proves that our proposed algorithm utilizes the motion information more efficiently, and confirms the capability of the estimated per-pixel optical flow. In Figure 7, rate-distortion (RD) curves of two sequences are presented, which demonstrate the effectiveness of the algorithm across a wide rate range. Moreover, it is observed that CLRF is used very often. For example, for city_cif,



(a) crowd_run_1080p



(b) blue sky 1080p

Fig. 7. RD curves of crowd_run (9.8% gain) and blue_sky (8.4% gain). Note the resulting PSNR range of blue_sky is relatively larger, therefore the difference of the two RD curves may appear smaller in the plot.

48.8% of the pixels belong to blocks that use CLRF as reference frame, further proving the effectiveness of our proposed scheme.

To highlight the impact on the performance due to the enhancements introduced in Section IV of this paper, we consider the gains they offered over our preliminary results that appeared in the conference paper [34], which is subsumed in Section III, and referred to herein as the *basic approach*. The basic approach yielded 2.3%, 2.8% and 2.7% BD-rate reduction for the low-res, mid-res and hd-res test sets, respectively, on top of the AV1 baseline. Thus, the additional enhancements of Section IV, improve the gains of the basic approach by a factor of over 1.3.

Recognizing the practical constraints of many video coding applications, the additional complexity of performing optical flow estimation (especially at the decoder) is an important consideration. As discussed in Section V, various parameters could influence the complexity of the proposed approach. First, as presented in Section V-A, the additional complexity should be linear in the the number of iterations for CG, M and M_b. This is confirmed by Figure 8, which clearly demonstrates such linear relationship for sequence *city_cif*. Note that, as also shown in the figure, even with the same number of iterations ($M = M_b$), the block-constrained algorithm yields a low complexity. This is because for the block-constrained algorithm, the relaxed condition at block boundaries effectively reduces

TABLE I
PEAK PERFORMANCE BD-RATE REDUCTION (%) OF THE PROPOSED METHOD

Low-res		Mid-res		HD-res	
akiyo	-2.97	BQMall	-4.45	basketballdrive	-3.021
basketballpass	-6.32	BasketballDrillText	-3.452	blue_sky	-8.414
blowingbubbles	-2.545	BasketballDrill	-4.207	bqterrace	-1.341
bowing	-3.778	Flowervase	-2.523	cactus	-6.173
bqsquare	-2.496	Keiba	-0.573	chinaspeed	-1.188
bridge_close	-0.566	Mobisode2	-2.789	city	-3.249
bridge_far	*0.133*	PartyScene	-2.65	crew	-3.801
bus	-2.693	RaceHorses	-4.262	*crowd_run*	*-9.764*
cheer	-2.591	aspen	-4.109	cyclists	-3.348
city	-5.667	city	-8.384	dinner	-3.624
coastguard	-1.037	controlled_burn	-1.665	ducks_take_off	-0.705
container	-2.006	crew	-5.377	factory	-4.162
crew	-3.927	*crowd_run*	*-8.984*	fourpeople	-3.925
deadline	-3.89	ducks_take_off	-1.058	in_to_tree	-3.892
flower	-2.362	harbour	-2.749	jets	-7.246
flowervase	-2.936	ice	-7.255	johnny	-2.154
football	-2.593	into_tree	-4.941	kimono1	-5.125
foreman	-4.804	old_town_cross	-6.728	kristenandsara	-3.673
garden	-2.268	park_joy	-3.132	life	-4.955
hallmonitor	-1.445	red_kayak	-0.178	mobcal	-0.645
harbour	-1.56	rush_field_cuts	-8.702	night	-4.355
highway	-1.919	sintel_trailer_2k	-1.217	old_town_cross	-4.212
husky	-1.631	*snow_mnt*	*-0.104*	parkjoy	-3.643
ice	*-7.751*	soccer	-3.05	parkrun	-1.238
keiba	-0.514	speed_bag	-1.141	parkscene	-5.201
mobile	-2.881	station2	-8.011	ped	-1.449
mobisode2	-2.326	tears_of_steel1	-2.84	riverbed	-0.215
motherdaughter	-5.759	tears_of_steel2	-5.127	rush_hour	-3.227
news	-3.045	touchdown_pass	-4.169	sheriff	-1.273
pamphlet	-3.403	west_wind_easy	-0.548	shields	-1.858
paris	-5.292			station2	-6.323
racehorses	-6.013			stockholm_ter	-2.395
signirene	-3.012			*sunflower*	*0.505*
silent	-3.101			tennis	-0.839
soccer	-2.416			tractor	-3.107
stefan	-1.699			vidyo1	-3.921
students	-4.401			vidyo3	-4.639
tempete	-1.18			vidyo4	-4.545
tennis	-2.234				
waterfall	-3.133				
Average	**-3.001**	**Average**	**-3.813**	**Average**	**-3.483**

the number of total non-zero elements in the sparse matrix by a constant factor. Furthermore, the decoder is able to skip blocks that are not referred to, thus further reducing the complexity. It should be noted that as the number of iterations increases, the complexity for the block-constrained algorithm eventually becomes a bit lower than the linear curve. This is due to the fact that we terminate the algorithm when a certain precision is reached, and that the block-constrained algorithm with fewer variables converges much faster. In our experiments, we noticed that such relationship and similar trends were also exhibited for other test sequences, and therefore for the rest of the section, we use the complexity of *city_cif* as a rough approximation of the average complexity.

Next, in Figure 9, the relationship between the coding performance gain for the low-res test set and additional decoder complexity, is presented (the number of iterations, M or M_b, is shown as the label of each data point). For both frame-based and block-constrained algorithms, the performance improves with increasing complexity until it saturates at a certain level. Also note that the block-constrained algorithm reaches the plateau much faster than the frame-based algorithm, but its maximal performance gain is lower, as it ignores the correlation across block boundaries. It can be concluded that, when the target decoder complexity is limited, the block-constrained algorithm serves as a good alternative in terms of the trade-off between complexity and overall performance.

Such trade-off is also influenced by other parameters. Table II presents the average coding gain for a few experiment sets, each with different choices of num_P, num_W and optical flow estimation algorithm (we fix $M_b = 10$ as it provides a better trade-off as shown in Figure 9). Obviously, by changing the parameters, different tradeoffs of performance and

TABLE II
TABLE OF BD-RATE REDUCTION (%) WITH VARIOUS COMPLEXITY TRADE-OFFS

	num_P	num_W	Frame-based	Block-based	Low-res	Mid-res	HD-res	Relative Complexity
set 1 (peak)	3	3	yes		-3.001	-3.813	-3.483	1
set 2	3	3		yes	-2.293	-3.056	-2.916	0.026
set 3	1	1	yes		-2.274	-2.955	-2.883	0.134
set 4	1	1		yes	-1.879	-2.663	-2.647	0.010
initialization only	0	0	-	-	-1.227	-2.076	-2.238	0

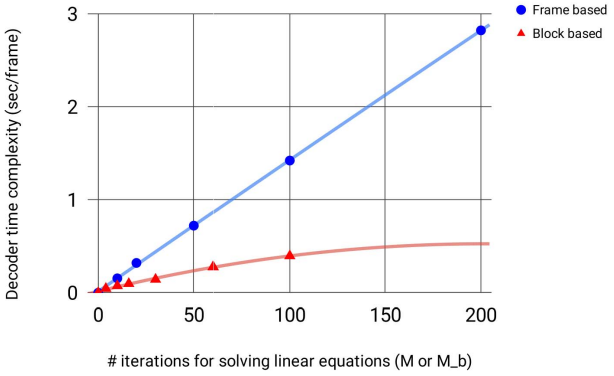


Fig. 8. The linear relationship of decoder time complexity v.s. number of iterations for solving linear equations (M or M_b).

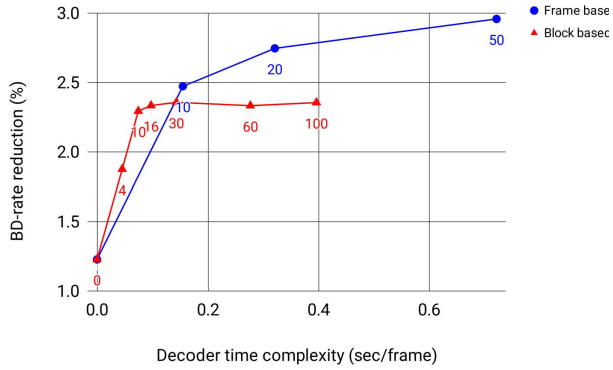


Fig. 9. Trade-off between performance and complexity for both frame-based and block-constrained algorithms. The label associated with each data point represents the number of iterations M or M_b.

complexity can be achieved. With the fastest setting (set 4), more than 60% of the coding gains are maintained, while the additional complexity is only 1% of that needed to achieve peak performance.

It is also worth noting that, for the "initialization only" set, we do not perform optical flow estimation at all, and use the initialization of motion field directly for CLRF generation (hence incurring nearly no additional complexity). As shown, it also yields a significant bit-rate reduction. This clearly shows the effectiveness of the motion vector initialization scheme, and how it lays a solid foundation for the subsequent optical flow estimation.

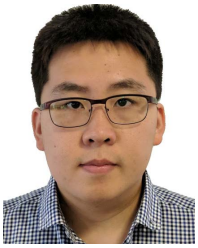
VII. CONCLUSION

This paper proposes a novel scheme for bi-directional motion compensated prediction that accounts for the motion information, already available to the decoder, between the two-sided reference frames. Optical flow estimation is utilized to provide a per-pixel motion field, capable of capturing non-translational motions. A co-located reference frame is interpolated according to the estimated motion field, and offset motion vectors are calculated and transmitted to correct possible deviation from the assumed motion model. The basic approach is complemented by various optimization techniques tailored to video codecs, including motion field initialization, motion vector prediction and confidence-based optical flow estimation. Moreover, a block-constrained algorithm is proposed, which is specifically designed for lower complexity applications. The effectiveness of the proposed scheme is evidenced by the experiment results, with significant BD-rate reductions across a large set of video sequences (3% to 4% on average). It is also shown that the proposed speed optimization incurs considerably lower complexity while maintaining most of the performance gains.

REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[2] D. Mukherjee *et al.*, "The latest open-source video codec VP9—An overview and preliminary results," in *Proc. Picture Coding Symp. (PCS)*, Dec. 2013, pp. 390–393.
[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[4] Y. Chen *et al.*, "An overview of core coding tools in the AV1 video codec," in *Proc. Picture Coding Symp. (PCS)*, Jun. 2018, pp. 24–27.
[5] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 4, pp. 419–422, Aug. 1996.
[6] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.
[7] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, and L.-G. Chen, "Survey on block matching motion estimation algorithms and architectures with new results," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 42, no. 3, pp. 297–320, Mar. 2006.
[8] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Trans. Image Process.*, vol. 3, no. 3, pp. 327–331, May 1994.
[9] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, nos. 1–3, pp. 185–203, Aug. 1981.
[10] D. Sun, S. Roth, and M. J. Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them," *Int. J. Comput. Vis.*, vol. 106, no. 2, pp. 115–137, Jan. 2014.
[11] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 500–513, Mar. 2011.
[12] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1744–1757, Sep. 2012.

[13] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2006, pp. 211–224.
[14] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1385–1392.
[15] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2758–2766.
[16] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2462–2470.
[17] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised deep learning for optical flow estimation," in *Proc. AAAI*, vol. 3, 2017, p. 7.
[18] F. Kendoul, I. Fantoni, and K. Nonami, "Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles," *Robot. Auto. Syst.*, vol. 57, nos. 6–7, pp. 591–602, Jun. 2009.
[19] A. G. Bors and I. Pitas, "Optical flow estimation and moving object segmentation based on median radial basis function network," *IEEE Trans. Image Process.*, vol. 7, no. 5, pp. 693–702, May 1998.
[20] R. Krishnamurthy, J. W. Woods, and P. Moulin, "Frame interpolation and bidirectional prediction of video using compactly encoded optical-flow fields and label fields," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 5, pp. 713–726, Aug. 1999.
[21] R. Krishnamurthy, P. Moulin, and J. Woods, "Optical flow techniques applied to video coding," in *Proc. Int. Conf. Image Process.*, vol. 1, 1995, pp. 570–573.
[22] S. Lin, Y. Q. Shi, and Y.-Q. Zhang, "An optical flow based motion compensation algorithm for very low bit-rate video coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 4, Dec. 1997, pp. 2869–2872.
[23] Y. M. Chi, T. D. Tran, and R. Etienne-Cummings, "Optical flow approximation of sub-pixel accurate block matching for video coding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, vol. 1, Apr. 2007, p. I-1017.
[24] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and C. S. Boon, "Inter frame coding with template matching spatio-temporal prediction," in *Proc. Int. Conf. Image Process. (ICIP)*, vol. 1, 2004, pp. 465–468.
[25] R. Wang, L. Huo, S. Ma, and W. Gao, "Combining template matching and block motion compensation for video coding," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst.*, Dec. 2010, pp. 1–4.
[26] S. Kamp and M. Wien, "Decoder-side motion vector derivation for block-based video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1732–1745, Dec. 2012.
[27] G. Venugopal, D. Marpe, and T. Wiegand, "Region-based template matching for decoder-side motion vector derivation," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2018, pp. 1–4.
[28] S. Klomp, M. Munderloh, Y. Vatis, and J. Ostermann, "Decoder-side block motion estimation for H.264 / MPEG-4 AVC based video coding," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2009, pp. 1641–1644.
[29] Y. Chin and C.-J. Tsai, "Dense true motion field compensation for video coding," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 1958–1961.
[30] S. Klomp, M. Munderloh, and J. Ostermann, "Decoder-side hierarchical motion estimation for dense vector fields," in *Proc. 28th Picture Coding Symp.*, Dec. 2010, pp. 362–365.
[31] A. Alshin, E. Alshina, and T. Lee, "Bi-directional optical flow for improving motion compensation," in *Proc. 28th Picture Coding Symp.*, Dec. 2010, pp. 422–425.
[32] A. Alshin and E. Alshina, "Bi-directional pptical flow for future video codec," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2016, pp. 83–90.
[33] J. Han, J. Feng, Y. Teng, Y. Xu, and J. Bankoski, "A motion vector entropy coding scheme based on motion field referencing for video compression," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 3618–3622.
[34] B. Li, J. Han, and Y. Xu, "Co-located reference frame interpolation using optical flow estimation for video compression," in *Proc. Data Compress. Conf.*, Mar. 2018, pp. 13–22.
[35] M. R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, vol. 49, no. 1. Washington, DC, USA: NBS, 1952.
[36] G. Bjontegaard, *Calcuation of Average PSNR Differences Between RD-Curves*, document VCEG-M33 ITU-T Q6/16, Austin, TX, USA, Apr. 2001, pp. 2–4.



Bohan Li (Member, IEEE) received the B.S. degree in electronics engineering from Tsinghua University in 2014, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Santa Barbara, in 2016 and 2019, respectively. He is currently a Software Engineer with the WebM team at Google. His main focus is on signal compression and video coding techniques. He is a member of the IEEE Signal Processing Society. He was a recipient of the Best Paper Award at the IEEE International Conference on Image Processing in 2017.

Jingning Han (Senior Member, IEEE) received the B.S. degree in electrical engineering from Tsinghua University in 2007, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California Santa Barbara in 2008 and 2012, respectively. He is currently a Senior Staff Engineer with Google. He is a main architect of the VP9 and AV1 codecs. His research interests include video coding and computer architecture. He held more than 50 U.S. patents in the field of video coding and published more than 60 research articles. He received the Dissertation Fellowship in 2012 from the Department of Electrical and Engineering, University of California Santa Barbara. He was a recipient of the Best Student Paper Award at the IEEE International Conference on Multimedia and Expo in 2012. He received the IEEE Signal Processing Society Best Young Author Paper award in 2015.

Yaowu Xu (Senior Member, IEEE) received the Ph.D. degree in nuclear engineering from Tsinghua University, Beijing, China, and the Ph.D. degree in electrical and computer engineering from the University of Rochester, Rochester, NY. He is currently a Principal Software Engineer with Google, leading Google's video compression team. His team is responsible for developing the core video technology that enables a broad range of products and services at Google including YouTube, Google Meet/Duo, Google Play, and Stadia. His team has been the driving force behind open source video compression technology VP9 and AV1. Prior to joining Google through its acquisition of On2 Technologies Inc., he was the Vice President of Codec Development at On2 Technologies, where he co-created the VPx series of video codecs from VP3 to VP8. He has been granted more than a hundred patents in the field of video compression.

Kenneth Rose (Fellow, IEEE) received the Ph.D. degree from the California Institute of Technology, Pasadena, in 1991. He then joined the Department of Electrical and Computer Engineering, University of California at Santa Barbara, where he is currently a Distinguished Professor. His main research activities are in the areas of information theory and signal processing, and include rate-distortion theory, source and source-channel coding, audio-video coding and networking, pattern recognition, and non-convex optimization. He has published over 350 peer-reviewed articles in these fields. A long-standing interest of his is in the relations between information theory, estimation theory, and statistical physics, and their potential impact on fundamental and practical problems in diverse disciplines. Recently, he was the senior coauthor of a article for which his students received the 2015 IEEE Signal Processing Society Young Author Best Paper Award. His earlier awards include the 1990 William R. Bennett Prize Paper Award of the IEEE Communications Society, as well as the 2004 and 2007 IEEE Signal Processing Society Best Paper Awards.