GLOBAL AND LOCALLY ADAPTIVE WARPED MOTION COMPENSATION IN VIDEO COMPRESSION

Sarah Parker[††], *Yue Chen*[††], *David Barker*[†], *Peter de Rivaz*[†], *Debargha Mukherjee*[††]

[††]Google Inc., Mountain View, USA
[†]Argon Design, Cambridge, UK

ABSTRACT

Real motion is often complex, and cannot always be sufficiently described by a translational model for the purpose of video coding. Translational motion models can only map a rectangular to a rectangular of the same size in a reference frame, but real motion observed in videos often go beyond that. For example, motion due to camera shake, panning and zoom might require transformations that support shearing, scaling, rotation and changes in aspect ratio. In order to handle these instances adequately for video encoding, we introduce two coding modes - a global motion mode and a locally adaptive warped motion mode, that make use of affine and/or homographic projections to more accurately describe and predict complex motion. These warped modes respectively capture global motion computed at the frame level, or local motion computed at the block level based on local motion statistics. We pay special attention to efficient SIMD friendly decoder-side warping operations to make these tools practically useful. Together, these techniques prove to yield considerable coding gains on standard test sets.

Index Terms— AV1, motion compensation, homography, warped motion, global motion

1. INTRODUCTION

Modern video codecs depend heavily on the ability to exploit temporal redundancies between video frames. This process often involves estimating the motion in regions of a frame over time and communicating the motion parameters in the bit-stream in order to enable a decoder to create a good predictor. Block motion compensation, a technique in which blocks resulting from a suitable partitioning of a frame are predicted from same-sized blocks in a previously encoded frame, has proven to be a simple yet effective approach. Here, a motion vector is used to indicate which block in a given reference frame should be used as a predictor for a block in the current frame. Modern codecs, such as VP9[1][2], H.264[3] and HEVC[4], all use this approach, where motion vectors used in motion compensation are translational only.

Real motion in video however, is rarely translation only. For example, camera shake from hand held cameras may be more accurately described as small rotations. Changes in aspect ratio due to both local object motion and camera motion may be better described as shearing. Panning and zooming are common in both consumer and professional content. The only way to get an accurate prediction with a purely translational motion model when the real motion is more complex, is to use very small prediction block sizes, which in turn makes prediction less efficient. In order to improve coding efficiency, we propose incorporating a set of new coding modes, that will allow complex motion to be captured at larger block sizes, by warping the blocks using an affine or perspective model.

Previous literature [5][6][7][8] has attempted to introduce homographies into video coding. Common implementations included handling global motion and local motion in separate stages. In some of these paradigms, the decoder warps an entire reference frame according to global motion parameters before performing local motion compensation. MPEG-4 Part 2 included a coding mode which applied global motion parameters to dynamic sprites [8]. However in later generation codecs H.264, HEVC, VP8, VP9 - all warping modes were dropped in favor of translational motion with more powerful sub-pel interpolation in conjunction with adaptive quad-tree like block partitioning. It was observed that the enhanced flexibility in these codecs in choosing prediction block sizes, along with powerful 6- or 8-tap separable subpel-interpolation can work just as well or better than warped motion modes. Warping operations, on the other hand, are regarded as substantially more complex than translational motion in software, because of the need to compute subpel locations in the reference frame based on the warp parameters, and the need for non-separable interpolation to get the predicted pixel values at those subpel locations. Hardware complexity is less of a concern. Furthermore, if the interpolation used with warp models is simply bilinear or bi-cubic, as in most prior work on warped motion, it will often produce worse prediction than translational motion using much better interpolation.

In this work, we reintroduce warped motion models within the framework of a state-of-the-art video codec and show that for certain video sequences with strong non-translational motion, it can in fact, still be substantially more efficient to code with warped motion than only translation. Specifically, we handle global and local warped motion separately in two different coding modes. The global motion coding tool is meant primarily for handling camera motion, and allows conveying motion models explicitly at the frame level for the motion between a current frame and any one or more of its references. The model may then be applied to any block within the frame to create a predictor from the corresponding reference, if there is an advantage in using them. The local warped motion coding tool aims to describe local motion implicitly, by deriving the model parameters at the block level using motion vectors from the causal neighborhood. Both coding modes compete in rate-distortion efficiency with other traditional translational modes, and are selected only if there is an advantage in using them. If either of these modes are used, the aforementioned affine warp mechanism is employed.

Additionally, we implement small affine warps by a horizontal shear followed by a vertical shear, with 8-tap interpolation filters being used for each shear, at a subpel precision much higher than what is typically used for translational motion. Two shears can be implemented efficiently in software using SIMD instructions.

Overall, our approach allows for the use of warps to capture complex motion without sacrificing decoder efficiency. The method was developed for consideration in the emerging AV1 codec from the Alliance for Open Media [9].

2. MOTION MODELS

Both the global and local warped motion coding tools we developed can use planar projective (also known as a *homography*) or affine transformations. If α represents a weight factor, (x, y) represents coordinates of an original pixel in a current frame, and (x', y') represents its warped coordinates in a reference frame, a full homography can be described by the following:

$$\alpha \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

The 3×3 homography matrix H above has 8 degrees of freedom and allows transforming a rectangle into an arbitrary quadrilateral. Starting from a homography, it is also possible to decrease the number of degrees of freedom, allowing the motion model to be encoded using fewer bits. Decreasing the degrees of freedom results in less complex, but often sufficient motion models that has the added benefit of being less noise-prone during estimation. Affine projections, which have only 6 degrees of freedom, allow a rectangular to parallelogram transformation. In this case H is restricted:

$$h_{31} = h_{32} = 0 \,,\; h_{33} = 1 \quad (2)$$

Affine projections support translation, rotation, scale, changes in aspect ratio, and shearing. Further, the warp can be implemented more efficiently (see Section 4) than full homographies. Similarity projections, which require 4 degrees of freedom, allow for a block to block transformation with rotation and zoom. In this case:

$$h_{31} = h_{32} = 0 \,,\; h_{33} = 1, \\ h_{11} = h_{22} \;,\; h_{13} = -h_{21} \quad (3)$$

Finally, simple translation, allows a block to block of the same size transformation with no rotation, and has 2 degrees of freedom. That is:

$$h_{31} = h_{32} = 0,\; h_{33} = 1, \\ h_{11} = h_{22} = 1,\; h_{12} = h_{21} = 0 \quad (4)$$

A trivial case is when further $h_{13} = h_{23} = 0$, or when the H matrix is identity. That corresponds to zero-motion.

In the following sections, we explore how these warp models are used in both the global and warped motion tools to improve block prediction.

3. MOTION MODEL COMPUTATION AND PREDICTION

The motion parameters for the global and local warped motion tools are computed and applied separately using different techniques. Here, we discuss the methods of parameter estimation and block prediction for both tools.

3.1. Global Motion

Each coded inter-frame has a set of pre-compressed reference frames from which block predictions may be made. When a frame is encoded, a set of global motion parameters is computed and transmitted between that frame and each reference frame. Subsequently, any block in the frame can signal use of the *global motion mode* with a given reference to create a suitable predictor.

While there are many ways to obtain global motion parameters between two frames, in the current implementation we use a feature matching scheme followed by robust model fitting. First, for each reference frame, FAST features [10] are computed in the current frame and the reference. FAST features are regions in the image that remain stable under changes in brightness and can be reliably tracked between frames. If enough interest points are found, we attempt to match points in the current frame to corresponding points in the reference frame. In this process, the normalized cross correlation is computed between all feature points in the two frames. For any point in the current frame, a point in the reference frame is determined to be a match if it fits the following criteria:

1. The point in the reference frame is within a predefined euclidean distance threshold.
2. The point in the reference frame has the highest correlation with the point in the current frame.

Once a set of correspondences is determined, it is used to compute a least-squares fit to a desired model using RANSAC [11]. RANSAC is an iterative algorithm for estimating model parameters from noisy data with outliers. RANSAC is invoked with a desired model type (translation, similarity, affine or full homography) to yield a set of corresponding model parameters that best fit the correspondence data. The parameters are then quantized and further refined using a refinement search within a small region in the quantized parameter space. Parameters that are ascertained as 'good' in terms of the warping error (obtained by warping the reference frame to the current frame and computing the error) are communicated at the frame level in the bitstream for that reference. Note that the best parameters computed may in fact be identity homography corresponding to zero-motion. Besides, parameters that are not 'good' are reverted to the identity homography to describe a default zero-motion model. Note that in the vast majority of static camera scenes we expect the global motion to be identity or zero-motion.

To avoid sending redundant bits, we iteratively try each model type starting from the lowest possible model, and working up to the highest. If a lower model already produces a good enough warping error, we terminate the search and use that model.

The quantized global motion parameters thus computed are sent at the frame level for each available reference. No matter what the model is, a block within a frame that signals use of the global motion mode with a given reference, would use the model with parameters indicated at the frame level for that reference.

3.2. Warped Motion

In the local warped motion tool, we estimate the warp parameters from translational motion vectors already conveyed in the current block and blocks in its causal neighborhood using the same reference frame. Since only translational motion vectors are encoded, and the decoder duplicates the process of obtaining the warp parameters from them, no overheads are needed to transmit the warp model parameters. At the block level, we simply signal the use of the *local warp model* if it is deemed most efficient.

The first step includes collecting pixel-to-pixel projection samples from the coded blocks adjacent to the current block. For each adjacent block that uses the same reference frame as the current block, the pixel in the center and its corresponding pixel location in the reference frame are used as a projection sample. For example, in Figure 1, say the motion of block i, $i = 0, 1, \ldots, 5$ is defined by motion vector $(u_x, u_y) = \boldsymbol{MV}_i$. Therefore a pixel (x, y) in block i is projected to

$$(x', y') = (x + u_x, y + u_y), \tag{5}$$

forming a projection sample, (x, y) - (x', y').

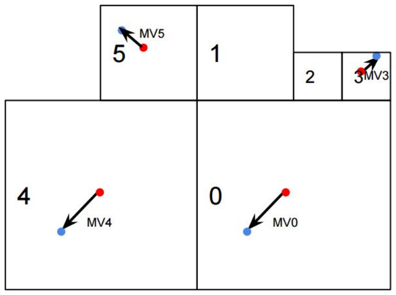

Figure 1: Illustration of projection samples

If the neighboring blocks choose global or local warped motion compensation rather than translational, projection samples are calculated according to the warped model used. Usually, one sample per block cannot produce enough stability for least square estimation, therefore 3 extra samples are created by shifting each original sample by a quarter pixel in one or two dimensions: +(0.25, 0), +(0, 0.25), and +(0.25, 0.25), are also accounted.

Once the projection samples have been computed, the parameters of the parametric image projection model are estimated from the projection samples by a least-squares minimization of the distance between the reference projection and the modeled projection. In order to maintain reasonable decoding complexity for local warped motion, we simplify the optimization algorithm by assuming the center pixel of the current block will be projected to the location described by the conveyed motion vector. Specifically, in the proposed framework where 6-parameter affine projection is supported, if we shift the origin of the source points to the block center, and the origin of the destination points to the block center added to the motion vector transmitted, we are able to ignore the translational terms and the projection model will be simplified as:

$$x' = h_{11}x + h_{12}y, \quad y' = h_{21}x + h_{22}y. \tag{6}$$

Therefore only 4 parameters need to be optimized. Further, these 4 parameters can be separated into two 2-dimensional least square problems that can be readily solved by the decoder using integerized math.

In the decoder, the motion mode for a block is set as a simple translational model by default. For single-reference inter blocks, we first count the number of blocks in the neighborhood including itself that use the same reference. If one or more blocks with the same reference are found, a motion mode symbol is decoded from the bitstream. If the *local warp mode* is signaled, the warp model is estimated. Once the warp model has been estimated, the current block is warped to a quadrilateral in the reference frame to yield the predictor. The actual warping operation used is the same for both the global and local warp motion tools. We describe that in detail in the next section.

4. BLOCK WARPING SCHEME

Even though the previous section describes full homographies, we find affine and similarity models to be sufficient in most cases. If we restrict ourselves to affine transformations only, it turns out that an efficient warping scheme can be obtained by applying two successive shear operations - one horizontal followed by another vertical. Specifically, an affine warp matrix can be factored as a product of two shear matrices as shown below:

$$\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \gamma & 1+\delta \end{bmatrix} \begin{bmatrix} 1+\alpha & \beta \\ 0 & 1 \end{bmatrix} \tag{7}$$

The second shear is applied by the horizontal filter, while the first shear is applied by the vertical filter. For each shear, interpolation is conducted with a powerful 8-tap subpel filter. The key advantage of this approach is that it allows the application of a separable filter to the entire block, rather than per-pixel. The warp process can be enumerated as follows:

1. Split up an input block into 8x8 blocks. For each block, we project the central point (4, 4) to get the overall block position.
2. Filter horizontally: Generate 15 rows of 8 pixels each by filtering horizontally with a 8-tap filter, where each pixel gets a different horizontal offset in 1/64th-pel precision.
3. Filter vertically: Generate the output 8x8 block by filtering vertically using a 8-tap filter, where each pixel gets a different vertical offset in 1/64th-pel precision.

In order to further improve efficiency, we have implemented this using SIMD, which processes 8-rows in parallel at a time. We have also restricted the region of pixels fetched from the reference frame to a 15x15 area to facilitate a fast hardware implementation. To accomplish this with an 8-tap filter, a trick is incorporated. In most cases, the fractional pixel offset relative to the displacement of the central pixel position in the 8x8 block will be in the [0, 1) range, in which case an 8-tap filter is easily applied. In cases where fractional pixel offsets go beyond the [0, 1) range, we instead use a 6-tap filter in order to stay within the 15x15 allotted region. In particular, if the offset is in the [-1, 0) range, a 6-tap interpolation filter is fit in the space of a 8-tap filter where the two taps on the right are zero. Likewise, if the offset is in the [1, 2) range, the same 6-tap interpolator is fit in the space of a 8-tap filter where the two filter taps on the left are 0. If the fractional offsets are less than -1, or more than 2, then the warp cannot be applied. Thus, one limitation of this approach is that the affine parameters must be sufficiently small. Since the horizontal filter generates 15 rows of 8 columns, and the initial point we project is at (4, 4) within the block, it can be shown that the parameters must satisfy:

$$4\,|\alpha| + 7|\beta| \leq 1 \quad \textit{and} \quad 4\,|\gamma| + 4\,|\delta| \leq 1 \tag{8}$$

Note that this is not a major restriction since usable motion parameters are almost always within this range, and the limits are almost always satisfied in our implementation.

5. BITSTREAM SYNTAX

5.1. Global Motion Syntax

The global motion information computed is added to the compressed header of each inter-frame. For each reference frame, the bitstream syntax includes an entropy coded flag to indicate which motion model is to be used among: zero-motion, translation, similarity, affine, or homography. For each model, the entropy coded flag is followed by the corresponding number of quantized parameters: 0, 2, 4, 6, or 8 depending on the model. Each parameter is conveyed using a fixed number of bits at a suitable precision.

Subsequently at the block level for an inter coded block, after the reference frame indices has been transmitted, a special *global* *motion mode* is indicated to invoke use of the global warped predictor.

5.2. Local Warped Motion Syntax

For an inter-coded block, after the reference frame index and motion vector has been transmitted, an additional symbol is added to indicate whether the *local warp motion mode* is to be used instead of the translational model. The symbol is only encoded if there is at least one block in the neighborhood that uses the same reference frame as the current block.

6. CODING RESULTS AND ANALYSIS

To evaluate our new tools, we perform a controlled bitrate test on the AV1 codebase using 3 different video sets:

1. *cam_lo*: 40 videos with a strong camera motion of 240p, CIF and SIF resolutions.
2. *lowres:* 40 videos of 240p, CIF and SIF resolutions.
3. *midres*: 30 videos of 4CIF and 480p resolution.
4. *hdres*: 38 videos at 720p and 1080p resolution.

150 frames of each video are coded with a single keyframe given a set of target bitrates.

For quality metrics we use average sequence PSNR and SSIM [12] computed by the arithmetic average of the combined frame PSNRs and SSIMs respectively. Combined frame PSNR is computed from the combined MSE of the Y, Cb and Cr components in a frame. In other words, assuming 4:2:0 sampling, we have:

$$MSE_{combined} = [4MSE_y + MSE_{Cb} + MSE_{Cr}]/6,$$

$$PSNR_{combined} = min(\ 10\log_{10}(255^2 / MSE_{combined}), 100\ ) \qquad (9)$$

SSIM for each component in each frame is computed by averaging the SSIM scores computed without applying a windowing function over 8×8 windows for each component. Combined SSIM for the frame is computed from the SSIMs of the Y, Cb and Cr components as follows:

$$SSIM_{combined} = 0.8\ SSIM_y + 0.1\ (SSIM_{Cb} + SSIM_{Cr}) \qquad (10)$$

To compare RD curves obtained by two codecs, we use a modified BDRATE [13] metric that uses piecewise cubic Hermite polynomial interpolation (*pchip*) on the rate-distortion points before integrating the difference over a finer grid.

Table 1 shows the average BDRATE reduction results for global warped motion mode only, local warped motion mode only, and combined warped modes for each video set. The warped modes are restricted to be at most affine in these tests. The baseline in all cases is AV1 with default configure options, while the configurations tested are with *--enable-global-motion* (global motion only), *--enable-warped-motion* (local warped motion only) and *--enable-global-motion --enable-warped-motion* (both global and local warp modes) respectively in the libaom AV1 codebase. As expected, the two warp modes do eat up each other's gains somewhat, yet the overall gains are non-trivial.

More importantly, most of the gains come from videos that have strong and sustained camera or complex local motion. Tables 2-4 show the BDRATE reduction results for selected individual videos for which these tools achieve considerable coding gains in the *lowres*, *midres* and *hdres* sets respectively. Most other videos in these sets remain either neutral within noise limits, or have limited gains. This is intuitive since videos without prominent camera motion or non-translational local motion cannot be expected to have any advantage from these tools.

As an aside, videos with strong parallax do not seem to get consistent gains from these modes, especially when only global motion mode is enabled. That is probably because a single affine motion model per reference frame may not be sufficient to accurately represent the motion in a scene when there is parallax. We are trying to address this issue in future work.

Table 1. AV1 average BDRATE results for global and local warped motion modes

Test Set	**Global warp only**		**Local warp only**		**Global+Local warp**	
	BDRATE (PSNR)	**BDRATE (SSIM)**	**BDRATE (PSNR)**	**BDRATE (SSIM)**	**BDRATE (PSNR)**	**BDRATE (SSIM)**
cam_lo	-3.31%	-4.00%	-2.50%	-3.08%	-4.86%	-5.79%
lowres	-1.53%	-2.07%	-1.03%	-1.25%	-2.22%	-2.80%
midres	-1.90%	-2.05%	-1.14%	-1.25%	-2.51%	-2.67%
hdres	-1.49%	-1.69%	-1.23%	-1.44%	-2.38%	-2.69%

Table 2. Selected BDRATE results on *lowres* set

Video	**BDRATE (PSNR)**	**BDRATE (SSIM)**
bqsquare_240p.y4m	-12.98%	-14.53%
flowervase_240p.y4m	-5.98%	-6.60%
tempete_cif.y4m	-12.41%	-17.74%
waterfall_cif.y4m	-15.14%	-20.98%

Table 3. Selected BDRATE results on *midres* set

Video	**BDRATE (PSNR)**	**BDRATE (SSIM)**
PartyScene_832x480_50.y4m	-4.77%	-5.57%
aspen_480p.y4m	-4.64%	-4.85%
into_tree_480p.y4m	-4.75%	-4.85%
station2_480p25.y4m	-37.99%	-41.30%

Table 4. Selected BDRATE results on *hdres* set

Video	**BDRATE (PSNR)**	**BDRATE (SSIM)**
blue_sky_1080p30.y4m	-10.03%	-13.75%
jets_720p30.y4m	-20.64%	-21.58%
mobcal_720p50.y4m	-7.07%	-7.13%
station2_1080p25.y4m	-22.09%	-26.56%

7. CONCLUSION

In this paper we have presented two tools designed to capture complex global and local motions within the framework of a state-of-the-art video codec. The proposed tools make special considerations for decoder complexity, particularly for software decoding. Results on varied test sets demonstrate their effectiveness. In future work, we plan to improve and extend the framework to cover multiple global motion models per reference frame.

8. REFERENCES

[1] D. Mukherjee, J. Bankoski, R. S. Bultje, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, "The latest open-source video codec VP9 - an overview and preliminary results," *Proc. IEEE Picture Coding Symp.*, pp. 390-93, San Jose, Dec. 2013.

[2] D. Mukherjee, J. Bankoski, R. S. Bultje, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, "A Technical Overview of VP9 - the latest open-source video codec," *SMPTE Motion Imaging Journal*, Jan/Feb 2015.

[3] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard; Ajay Luthra. "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13 No. 7, Jan 2011.

[4] Gary J. Sullivan, JensRainer Ohm, WooJin Han, and Thomas Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 22, No. 12, Dec 2012.

[5] Dufaux, Frederic, and Janusz Konrad. "Efficient, robust, and fast global motion estimation for video coding." IEEE transactions on image processing 9.3 (2000): 497-501.

[6] Tse, Y. Tong, and Richard L. Baker. "Global zoom/pan estimation and compensation for video compression." Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on. IEEE, 1991.

[7] Jozawa, Hirohisa, et al. "Two-stage motion compensation using adaptive global MC and local affine MC." IEEE Transactions on Circuits and Systems for video technology 7.1 (1997): 75-85.

[8] Watanabe, Hiroshi, and Kumi Jinzenji. "Sprite coding in object-based video coding standard: MPEG-4." Proceedings of Multiconference on Systemics, Cybernetics and Informatics. Vol. 13. 2001.

[9] Alliance for Open Media, http://aomedia.org/

[10] Rosten, Edward, and Tom Drummond. "Fusing points and lines for high performance tracking." *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 2. IEEE, 2005.

[11] Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM* 24.6 (1981): 381-395.

[12] Wang, Zhou; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. (2004-04-01). "Image quality assessment: from error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, No. 4, pp. 600–612, April 2004.

[13] G. Bjøntegaard, "Calculation of average psnr differences between rdcurves," VCEGM33, 13th VCEG meeting, Austin, Texas, March 2001.