ADAPTIVE INTERPOLATION FILTER SCHEME IN AV1

Ching-Han Chiang, Jingning Han, Stan Vitvitskyy, Debargha Mukherjee, and Yaowu Xu

WebM Codec Team, Google Inc.
1600 Amphitheatre Parkway, Mountain View, CA 94043
Emails: {angiebird,jingning,vitvitskyy,debargha,yaowu}@google.com

ABSTRACT

Video codecs heavily depend on sub-pixel level motion compensation to achieve superior compression performance. Interpolation filters with both anti-aliasing and denoising properties play a critical role in producing high quality prediction at sub-pixel positions. Prior research has developed many adaptive filtering schemes to improve the prediction precision for compression gains. On the other hand, such filtering operations require intense computation and may lead to scattered cache footprints, therefore, account for a major portion of the overall decoding cost in both software and hardware implementations. An adaptive interpolation filtering scheme is proposed in this work to optimize the trade off between prediction quality and decoding performance. It employs a separable model and selects filter kernels independently for horizontal and vertical directions to better capture statistical variations. In order to obtain sharper transition and reduce the ripple effect in the passband in frequency domain, a 12-tap filter is introduced in conjunction with a complimentary operation design that minimizes its impact on the decoding performance. The scheme achieves on average 1.3% coding gains across a wide range of test settings, with fairly limited additional hardware cost.

Index Terms— Interpolation filter, adaptive filter, motion compensated prediction

1. INTRODUCTION

Sub-pixel level motion compensated prediction is widely used in video compression. A common strategy is to use the interpolation filter to generate sub-pixel reference from previously coded pixel frames. In addition to the anti-aliasing property, these interpolation filters may often provide denoising functionality for prediction accuracy. The noise source includes acquisition noise in the original signal and the quantization noise in the previously reconstructed frames. It has been demonstrated that the temporal correlation in the frequency domain varies significantly across video signal [1]. Clearly this would require certain flexibilities of the interpolation filter such that it could adapt to those statistical variations for prediction quality.

A non-separable adaptive filter design is proposed in [2], where the filter coefficients associated with a large set of surrounding pixels are determined by solving a linear minimization problem. The resulting coefficients are explicitly sent to the decoder to reproduce the predictor. This has been extended to the context of separable filter in [3]. A directional adaptive filter is further developed in [4].

Alternatively one can support a set of pre-defined filter kernels with distinctive frequency responses. A coding block finds the one that minimizes the rate-distortion cost and sends its index to the decoder to reproduce the prediction. The VP9 codec [5] employs such approach, where three interpolation filter types, namely SMOOTH, SHARP and NORMAL, are supported at coding block level. Each filter type corresponds to a linear low-pass filter with certain passband and cutoff frequency. The selected filter type (kernel) is then applied to both vertical and horizontal directions to build sub-pixel motion compensated reference.

Part of this work premises on the realization that the source signal may possess different characteristics in vertical and horizontal directions, which incurs the need for different filter responses to capture the disparity in temporal correlations. It allows each direction to select its own filter type independently, the combination of which provides more flexibility in the 2-D space to approximate the desired filter property.

When the source signal contains strong high frequency components, a wider passband filter (i.e. SHARP) is typically involved for interpolation. To reduce the ripple effect in the passband and obtain a sharp transition at cutoff frequency, a higher order linear filter is preferred. Due to the hardware constraints, however both VP9 [5] and HEVC [6] employ up to 8-tap filter. In a conventional scheme, where a single filter type is applied to both vertical and horizontal directions, a filter of order above 8 (and below 16) would require a substantial amount of depth-4 ($\log 16$) adder trees, which could potentially double the silicon area for a hardware decoder. See Section 5 for the breakdowns. This work overcomes such intricacy by introducing a constrained separable filter scheme, where up to one side is allowed to use 12-tap filter. To produce a 2-D sub-pixel reference block with a hybrid 8-/12-tap filter types, the predictor always processes the direction with shorter filter first, followed by the longer filter direction in the second stage. As evidenced in later section, this design significantly reduces the amount of deep adder trees in hardware decoder. It hence allows the use of 12-tap filter for wide passband interpolation.

The proposed interpolation filter scheme is implemented in the AV1 codec, a successor of VP9 joint developed by the Alliance of Open Media [7]. It is experimentally shown that it provides consistent compression performance gains over a large set of test conditions, with limited increment in hardware complexity. This paper provides an overview of the interpolation filter system in AV1 and its design principle.

2. INTERPOLATION OPERATION

The motion compensated prediction system in AV1 supports four interpolation filter types with precision up to $\frac{1}{8}$ sub-pixel for luma signal and precision up to $\frac{1}{16}$ for chroma signal. Each filter type can be represented by a filter kernel $F : 8 \times taps$ (or $16 \times taps$), where 8 (or 16) represents the number of phase offsets and $taps$ is the order of linear filter. The interpolation is performed by 2 one-dimensional filters, one for horizontal direction and one for vertical direction.

Given a filter kernel F associated with a filter type and a sub-pixel position (s_x, s_y) derived from the motion vector, a prediction block at the sub-pixel position can be generated by the following two steps. First, an intermediate block M is generated by applying the horizontal filter $F[s_x]$ to the corresponding integer position block V in the previously reconstructed frame,

$$M[x][y] = V[x + \frac{s_x}{8}][y] = \sum_{k=0}^{taps} V[x - fo + k][y] \times F[s_x][k], \quad (1)$$

where $fo = \frac{taps}{2} - 1$,

Then the intermediate block M undergoes a vertical filter $F[s_y]$ to produce the final prediction block P,

$$P[x][y] = V[x + \frac{s_x}{8}][y + \frac{s_y}{8}] = \sum_{k=0}^{taps} M[x][y - fo + k] \times F[s_y][k]. \quad (2)$$

3. INTERPOLATION FILTER TYPES

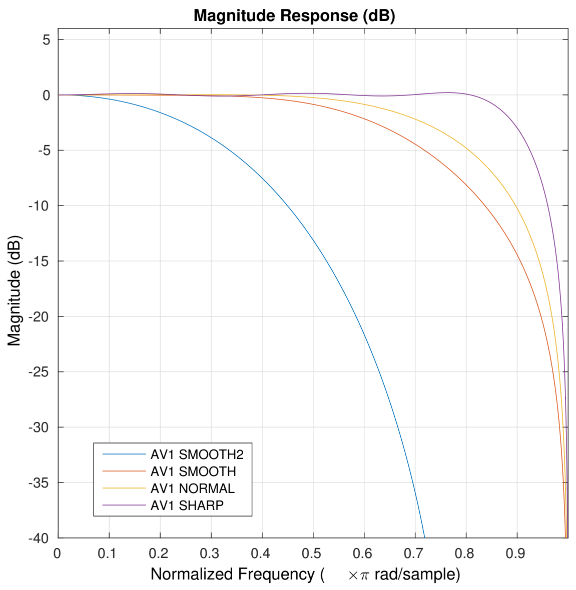

Fig. 1. The amplitude of frequency response of the 4 filter types in AV1 at half-pixel position.

A main challenge of interpolation filter design is to handle a wide varieties of temporal correlation in frequency domain due to noise or distortion effect. In this work, four candidate filter types are provided to cover different passband characteristics. It reserves the most commonly used NORMAL filter in VP9, redesigns the SMOOTH and SHARP filters, and introduces an additional SMOOTH2 filter. The new SHARP filter now employs a 12-tap kernel and hence has less ripples in the passband and sharper transition at cutoff frequency comparing to the SHARP filter in VP9. All the rest filter types remains to use 8-tap kernel.

In addition, the AV1 increases the cutoff frequency of SMOOTH filter; and adds a second SMOOTH2 filter with significantly lower cutoff frequency. The half-pixel frequency amplitude response of each filter type is shown in Fig. 1 and the design details are discussed as follows.

Filter Type	Coefficient
SHARP	(-1, 3, -6, 12, -24, 80, 80, -24, 12, -6, 3, -1)
REGULAR	(-1, 6, -19, 78, 78, -19, 6, -1)
SMOOTH	(1, -2, -7, 72, 72, -7, -2, 1)
SMOOTH2	(-1, 2, 20, 43, 43, 20, 2, -1)

Table 1. Filter coefficients at half pixel offset.

The SHARP filter is designed for the prediction block with high signal bandwidth and relatively low noise level. For each sub-pixel position, a 12-tap filter is used to satisfy the demand of flat passband and sharp transition at cutoff frequency. The filter coefficients is generated as least-squares linear-phase FIR filter, to have no decay factor for signal within 0 to 0.85 of Nyquist frequency.

The SMOOTH filter is designed for the prediction block with lower signal bandwidth and larger noise/distortion. To put more emphasis on decay the high frequency noise or distortion, a Hamming window-based design method is applied. For each sub-pixel position, an 8-tap filter is generated by setting the Hamming window parameter to 0.75.

The SMOOTH2 filter is designed for the prediction block whose motion vector does not reflect the real motion. Therefore, SMOOTH2 filter performs like averaging the integer pixels around the sub-pixel positions. For each sub-pixel position, an 8-tap filter is generated by setting the Hamming window parameter to 0.35.

To ensure computation efficiency, the filter is implemented as 7-bit integer arithmetics. The floating point filter coefficients are translated into a fixed-point representation, each coefficient is rounded as a 7-bit integer. Moreover, to guarantee that each sub-pixel position has unit transfer gain, the fixed-point filter coefficients are further slightly tuned such that their sum is equal to 2^7. The filter coefficients at half pixel position are shown in Table 1.

4. DUAL INTERPOLATION FILTER KERNEL

As discussed above, the source video signal may have distinctive statistics in vertical and horizontal directions, which in conjunction with the acquisition noise and quantization distortion would potentially produce different correlations along the two directions. To better capture such disparity, a dual filter kernel selection is allowed in AV1 for motion vectors that require sub-pixel offset in both directions. Each direction can select one of the above four filter type candidates independently. That makes a total of 16 possible filter responses in the 2-D space.

The vertical and horizontal filter types are coded independently in the bit-stream. When the motion vector appears to have only one component in sub-pixel position, it will skip coding the filter type of the full-pixel direction, since no interpolation process is needed therein. The probability model used to code the directional filter type is conditioned on the above and left neighbors' filter type in the same direction.

5. CONSTRAINED DUAL FILTER SYSTEM FOR HARDWARE EFFICIENCY

A natural concern about the use of 12-tap filter rises in its significant cost in terms of chip area and latency in hardware design. The hardware must be designed for the worst (i.e., most complex) scenarios.

8-8-tap cost	
1st stage multi	(7 + 4) * 4 * 8 = 352
2nd stage multi	4 * 4 * 8 = 128
1st stage adder tree	(7 + 4) * 4 = 44 depth-3
2nd stage adder tree	(4 * 4) = 16 depth-3
12-12-tap cost	
1st stage multi	(11 + 4) * 4 * 12 = 720
2nd stage multi	4 * 4 * 12 = 192
1st stage adder tree	(11 + 4) * 4 = 60 depth-4
2nd stage adder tree	(4 * 4) = 16 depth-4
8-12- tap or 12-8-tap cost	
1st stage multi	(11 + 4) * 4 * 8 = 480
2nd stage multi	4 * 4 * 12 = 192
1st stage adder tree	(11 + 4) * 4 = 60 depth-3
2nd stage adder tree	(4 * 4) = 16 depth-4

Table 2. Comparison of the numbers of multipliers and adder trees.

In the settings of interpolation filter, it occurs at 4x4 block. Let's consider the number of multiplications first. As summarized in Table 2, using two 8-tap filters in horizontal and vertical directions to produce a 4x4 block will take a total of 480 multiplications. On the other hand, applying two 12-tap filters will take 912 multiplications. The number of multiplications nearly doubles from 8-tap to 12-tap filter. We next consider additions. For an 8-tap filter, each output is generated by 7 additions, which is implemented as a depth-3 adder tree. Applying 8-tap filters in two directions will need 60 depth-3 adder trees. In contrast, a 12-tap filter needs 11 additions, which translates into a depth-4 adder trees. Hence, applying 12-tap filters on horizontal and vertical directions will require 76 depth-4 adder tree. Such increase in adder tree depth makes the latency in each addition stage up by 33.3%, which significantly increases the design area due to the necessity of using faster (and larger) adders.

To alleviate the additional cost of multipliers and adder trees, this work proposes a constrained dual filter approach, where the vertical and horizontal directions will use the 12-tap SHARP filter exclusively. When both the vertical and horizontal directions select SHARP filter, we arbitrarily set one of them to use 8-tap SHARP filter and the other to use 12-tap SHARP filter. In case a prediction block has one direction using 12-tap filter and a second direction using 8-tap filter, the predictor always processes the one with shorter filter first, followed by the longer filter direction as the second stage. This would reduce the number of multiplications for 4x4 block down to 672. Moreover, only 16 depth-4 adder trees are needed in the second stage, which alleviates the needs of design area for faster (and larger) adders.

6. EXPERIMENTAL RESULTS

We implement the proposed interpolation filter scheme in the AV1 framework and test it on low-resolution (CIF, SIF) and high-resolution (720P, 1080P, XGA) datasets. As compared to the interpolation filter system in VP9 [5], redesigning the SMOOTH and SHARP filters and introducing additional SMOOTH2 filter provide 0.886% compression performance gains on low-resolution dataset and 0.520% on high-resolution dataset in terms of Bjntegaard delta rate (BDRate) reduction. The dual filter system that allows each direction to select its own filter kernel further improves the coding performance by 0.522% and 0.864% in BDRate reduction for low-resolution and high-resolution datasets respectively.

The exclusive usage of 12-tap SHARP filter constraint significantly reduces the hardware latency and chip area cost, at the expense of a slight drop in coding performance – 0.107% and 0.015% in terms of BDRate reduction for low-resolution and high-resolution datasets respectively.

Overall, the proposed interpolation filter outperforms that of VP9 by 1.299% and 1.365% for low-resolution and high-resolution datasets in terms of BDRate reduction. The coding gain breakdown is shown in Table 3-4.

7. CONCLUSIONS

An interpolation filter scheme is proposed to provide more flexibility to capture signal variations in frequency domain. A complementary strategy that optimize the trade off between compression performance and hardware cost is presented. The scheme provides consistent compression performance gains over a wide range of test settings.

Table 3. Coding gains of the proposed interpolation filter scheme over the VP9 interpolation filter design on low-resolution dataset.

	res	BDRate(%)
akiyo	CIF	-2.495
basketballpass	240p	-1.2
blowingbubbles	240p	-1.281
bowing	CIF	-0.108
bqsquare	240p	-3.319
bridge_close	CIF	-0.266
bridge_far	CIF	-2.219
bus	CIF	-1.784
cheer	SIF	-0.267
city	CIF	-2.03
coastguard	CIF	-1.704
container	CIF	-2.04
crew	CIF	-1.021
deadline	CIF	-1.125
flower	CIF	-0.645
flowervase	240p	-2.633
football	CIF	-0.783
foreman	CIF	-0.84
garden	SIF	-0.582
hallmonitor	CIF	-1.101
harbour	CIF	-1.891
highway	CIF	-1.264
husky	CIF	-0.206
ice	CIF	-0.623
keiba	240p	-1.602
mobile	CIF	-2.489
mobisode2	240p	-0.891
motherdaughter	CIF	-0.998
news	CIF	-1.451
pamphlet	CIF	-0.325
paris	CIF	-1.067
racehorses	240p	-0.739
signirene	CIF	-0.692
silent	CIF	-0.555
soccer	CIF	-3.289
stefan	SIF	-0.391

students	CIF	-1.674
tempete	CIF	-1.572
tennis	SIF	-0.368
waterfall	CIF	-2.417
OVERALL		-1.299

Table 4. Coding gains of the proposed interpolation filter scheme over the VP9 interpolation filter design on high-resolution dataset.

	res	BDRate(%)
basketballdrive	1080p	-3.252
blue_sky	1080p	-1.19
bqterrace	1080p	-2.718
cactus	1080p	-1.168
chinaspeed	XGA	0.15
city	720p	-2.114
crew	720p	-0.941
crowd_run	1080p	-0.268
cyclists	720p	-1.458
dinner	1080p	-0.685
ducks_take_off	1080p	-2.016
factory	1080p	-0.566
fourpeople	720p	-1.657
in_to_tree	1080p	-1.645
jets	720p	-1.201
johnny	720p	-2.074
kimono1	1080p	-0.078
kristenandsara	720p	-1.324
life	1080p	-0.808
mobcal	720p	-2.914
night	720p	-0.733
old_town_cross	720p	-2.085
parkjoy	1080p	-0.466
parkrun	720p	-1.571
parkscene	1080p	-0.649
ped	1080p	-0.691
riverbed	1080p	-0.181
rush_hour	1080p	-0.429
sheriff	720p	-1.588
shields	720p	-2.372
station2	1080p	-1.556
stockholm_ter	720p	-2.165
sunflower	720p	-1.957
tennis	1080p	-0.64
tractor	1080p	-2.766
vidyo1	720p	-1.299
vidyo3	720p	-1.514
vidyo4	720p	-1.297
OVERALL		-1.365

8. REFERENCES

[1] Jingning Han, Vinay Melkote, and Kenneth Rose, "Transform-domain temporal prediction in video coding with spatially adaptive spectral correlations," in *IEEE Multimedia Signal Processing Workshop (MMSP)*, 2011.

[2] Yuri Vatis, Bernd Edler, Dieu Thanh Nguyen, and Jörn Ostermann, "Motion-and aliasing-compensated prediction using a two-dimensional non-separable adaptive wiener interpolation filter," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. IEEE, 2005, vol. 2, pp. 891–894.

[3] Steffen Wittmann and Thomas Wedi, "Separable adaptive interpolation filter for video coding," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE, 2008, pp. 2500–2503.

[4] Dmytro Rusanovskyy, Kemal Ugur, Antti Hallapuro, Jani Lainema, and Moncef Gabbouj, "Video coding with low-complexity directional adaptive interpolation filters," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 8, pp. 1239–1243, 2009.

[5] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu, "A technical overview of vp9 - the latest open-source video codec," *SMPTE*, vol. 2013, no. 10, pp. 1–17, 2013.

[6] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding HEVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[7] "Alliance of open media - source code repository," *https://aomedia.googlesource.com/*.