A SWITCHABLE LOOP-RESTORATION WITH SIDE-INFORMATION FRAMEWORK FOR THE EMERGING AV1 VIDEO CODEC

Debargha Mukherjee[¶], *Shunyao Li*[†], *Yue Chen*[¶], *Aamir Anis*[‡], *Sarah Parker*[¶], *James Bankoski*[¶]

[¶] Google, Inc., Mountain View, California
[†] University of California, Santa Barbara, California
[‡] University of Southern California, Los Angeles, California

ABSTRACT

Image restoration schemes have traditionally been targeted only for use in a blind scenario, where the aim is to improve the quality of an image or video after it has suffered degradations in capture, processing, storage, coding or transmission, at a time when the undegraded source is no longer available. However, these schemes can also be adopted for compression, because any information that can be restored can be saved, thereby aiding compressibility. In this use case, the source is known at the time of compression, and the encoder can send additional side-information to the decoder with the compressed bit-stream to specify how the decoder is supposed to restore. In this paper, we describe three novel schemes that are applied in-loop to reconstructed frames after a conventional deblocking loop filter has been applied. These schemes are switchable within a frame per suitably sized tile. The specific schemes described are based on separable symmetric Wiener filters, dual self-guided filters with subspace projection, and domain transform recursive filters. Results with the new coding tool are presented on the emerging AV1 video codec on standard test sets, showing upwards of 2% bit-rate savings.

Index Terms— Image restoration, deblocking, deblurring, edge-preserving smoothing, bilateral filters, guided filters, domain transform filters, Wiener filters.

1. INTRODUCTION

Image restoration [1]-[4] is a relatively mature field with many sophisticated techniques available for deblocking, deblurring, deringing, debanding, denoising, contrast enhancement, sharpening and resolution enhancement. Most of them are used in scenarios where the undegraded source is no longer available. Recently there has been increasing focus on use of such methods [5]-[9] for image and video codecs as well, with an aim to partially recover the information loss from traditional compression processes, and/or make the decoded image or video look visually more pleasing. One type of restoration that has been used in video codecs [10]-[14] over multiple generations is deblocking, which is really critical given the block based nature of modern hybrid video codecs. In latest generation video codecs, deblocking is conducted in the prediction loop, so that the restored frames can be used as references for coding subsequent frames. Even though several in-loop restoration tools beyond deblocking such as edge-preserving denoising or enhancements have shown promise in the literature [8][9], incorporation in standardized or commercial video codecs has been limited. One notable exception is the Sample Adaptive Offset (SAO) [10] tool in the latest standardized codec HEVC [12], that adds offsets to certain pixels in a deblocked frame to correct for systematic drifts. Recently methods [5][6] have been proposed for royalty-free codecs Daala and NetVC.

One reason why incorporation of sophisticated restoration schemes has been particularly difficult within video codecs is the substantial complexity involved. Most image restoration schemes in the literature are too complex to incorporate in a video codec where the expectation is to decode a bitstream often carrying high resolution content at up to 60 frames per second on computationally constrained devices. On the other hand, given the non-blind nature of the compression problem, and the ability to send side-information from the encoder to the decoder, innovative ways of adopting restoration paradigms are possible to substantially ameliorate the complexity issue.

In this spirit, we propose a set of in-loop restoration schemes for use in video coding post deblocking, to generally denoise and enhance the quality of edges, beyond the traditional deblocking operation. Because content statistics can vary substantially within a frame, these tools are integrated within a switchable framework where different tools can be triggered in different regions of the frame. Special attention is paid to decoding complexity. Our method was developed for consideration in the emerging AV1 codec from the Alliance for Open Media, and coding results are presented on the AV1 codebase.

Finally note that the coding architecture we are dealing with here can be viewed as a form of SNR scalable coding, but the tools involved are based on image restoration, and the side-information transmitted comprises a very small proportion of the overall bits.

2. RESTORATION TOOLS

In this section we first describe the essential restoration tools that are used in the switchable framework.

2.1 Separable Symmetric Wiener Filter

One restoration tool that has been shown to be promising in the literature is the Wiener filter [7][8]. Every pixel in a degraded frame could be reconstructed as a non-causal filtered version of the pixels within a $w \times w$ window around it where $w = 2r + 1$ is odd for integer r. If the 2D filter taps are denoted by a $w^2 \times 1$ element vector F in column-vectorized form, a straightforward LMMSE optimization leads to filter parameters being given by:

$$F = H^{-1}M \tag{1}$$

where $H = \mathrm{E}[XX^T]$ is the autocovariance of x, the column-vectorized version of the w^2 samples in the $w \times w$ window around a pixel, and $M = E[YX^T]$ is the cross correlation of x with the scalar source sample y, to be estimated. The encoder can estimate H and M from realizations in the deblocked frame and the source, and send the resultant filter F to the decoder. However, that would not only incur a substantial bit rate cost in transmitting w^2

taps, but also non-separable filtering will make decoding prohibitively complex. Therefore we impose several additional constraints on the nature of F. First, we constrain F to be separable so that the filtering can be implemented as separable horizontal and vertical w-tap convolutions. Second, we constrain each of the horizontal and vertical filters to be symmetric. Third, we assume the sum of both the horizontal and vertical filter coefficients to sum to 1. With these constraints, note that for each filter only r values need to be sent from the encoder to the decoder. Mathematically, the $w^2 \times 1$ element vector $F = column\_vectorize[ab^T]$ where a and b are $w \times 1$ vertical and horizontal filters s.t.: $a(i) = a(w-1-i)$, $b(i) = b(w-1-i)$, for $i = 0, 1, \ldots, r-1$, and $\sum a(i) = \sum b(i) = 1$.

The filter optimization with these constraints follow an iterative scheme where starting from an initial guess of the horizontal and vertical filters, one of them is optimized while holding the other fixed. We have seen very minor difference in coding performance with these constraints imposed over the unconstrained non-separable filter. We skip the derivation in the interest of space and only provide the equations for these two steps. Let

$$P = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & 1 \\ -2 & -2 & \ldots & -2 \\ \ldots & \ldots & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 1 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \end{bmatrix} \quad (2)$$

be a size $w \times r$ mixing matrix. Also, let H and M and be be decomposed as:

$$H = \begin{bmatrix} H_{00} & H_{01} & \ldots & H_{0,w-1} \\ H_{10} & H_{11} & \ldots & H_{1,w-1} \\ \ldots & \ldots & \ldots & \ldots \\ H_{w-1,0} & H_{w-1,1} & \ldots & H_{w-1,w-1} \end{bmatrix} \quad (3)$$

$$M = [\, M_0 \; M_1 \; \ldots \; M_{w-1} \,] \quad (4)$$

where H_{ij} and M_i are block matrices of size $w \times w$ and $1 \times w$ respectively.

Update a while keeping b fixed:

1. Compute: $U = \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} b(i)\, b(j)\, H_{ij} P$ which is $w \times r$.
2. Compute $z = \sum_{i=0}^{w-1} b(i)\, M_i P - U_r$ where U_r is the rth row of U. Note z is $1 \times r$.
3. Solve: $\hat{a}^T = z \,.\, (P^T U)^{-1}$ where $\hat{a}^T = [\, a(0)\; a(1) \; \ldots \; a(r-1) \,]$ is the first r samples of a, while the rest are given by the normalization and symmetry constraints, i,.e. $a = P\,\hat{a} + Z_r$, with $Z_r = [\, 0 \ldots 1 \ldots 0 \,]$ a $1 \times w$ row-vector with the rth element 1 and the rest 0.

Update b while keeping a fixed:

1. Compute $w \times w$ matrix V s.t. the (i,j)th element $V_{ij} = a^T H_{ij}\, a$, and then $U = VP$, which is $w \times r$.
2. Compute $1 \times w$ row vector t s.t. the ith element $t_i = M_i\, a$, and then $z = tP - U_r$ where U_r is the rth row of U, and z is $1 \times r$.
3. Solve: $\hat{b}^T = z \,.\, (P^T U)^{-1}$, where $\hat{b}^T = [\, b(0)\; b(1) \; \ldots \; b(r-1) \,]$ is the first r samples of b, while the rest are given by the normalization and symmetry constraints, i.e. $b = P\,\hat{b} + Z_r$.

In our implementation $w = 7$, which means 3 taps need to be sent for each filter. We use 4, 5 and 6 bits respectively for the first three filter taps while the others are obtained automatically from the normalization and symmetry constraints. A total of 30 bits are thus sent for both filters. Somewhat better results can be expected with larger w, but we limit to 7 in the interest of complexity.

2.2 Dual Self-guided Filtering with Subspace Projection

Guided filtering [3] is one of the more recent paradigms of image filtering where a fairly simple local linear model:

$$y = F\,x + G \quad (5)$$

is used to compute the filtered output y from an unfiltered sample x, where F and G are determined based on the statistics of the degraded image and a guidance image in the neighborhood of the filtered pixel. If the guide image is the same as the degraded image, the resultant so-called self-guided filtering has the effect of edge preserving smoothing.

The specific form of self-guided filtering we propose depends on two parameters: a radius r and a noise parameter e, and is enumerated as follows:

1. Obtain mean μ and variance σ^2 of pixels in a $(2r+1) \times (2r+1)$ window around every pixel. This can be implemented efficiently with box filtering based on integral imaging.
2. Compute for every pixel:
$$f = \sigma^2/(\sigma^2 + e);\; g = (1 - f)\mu \quad (6)$$
3. Compute F and G for every pixel as averages of f and g values in a 3×3 window around the pixel for use in (2).

Filtering is controlled by r and e, where a higher r implies a higher spatial variance and a higher e implies a higher range variance.

It turns out that such a cheap filtering operation with a single pair of parameters is not sufficient to get a degraded image close to the source. Therefore what we propose is to first produce two cheap restored versions with two sets of $\{r, e\}$ parameters, followed by projecting the difference between the source and the degraded frame, on the subspace generated by the difference between the two restored versions and the degraded frame. Specifically, if the degraded source is denoted X in column vector form, we first generate a restored version X_1 with parameters $\{r_1, e_1\}$ and then a second restored version X_2 with parameters $\{r_2, e_2\}$, and lastly the final restored version X_r is obtained as:

$$X_r = X + \alpha.(X_1 - X) + \beta.(X_2 - X) \quad (7)$$

Given X, X_1, X_2, and the source Y, the encoder can readily compute the α, β parameters using:

$$\{\alpha,\, \beta\}^T = (A^T A)^{-1} A^T b \quad (8)$$

where $A = \{\, X_1 - X,\; X_2 - X \,\}$, and $b = Y - X$.

The encoder sends the 6-tuple $\{\, r_1,\, e_1,\, r_2,\, e_2,\, \alpha,\, \beta \,\}$ to the decoder, which enables it to obtain first the cheap restored versions followed by compositing the final restoration. We get good results for small but different values of the radius parameters r_1 and r_2.

The parameters $\{r_1, e_1, r_2, e_2\}$ are taken from a small codebook. The multipliers $\{\, \alpha, \beta \,\}$ need much higher precision to produce good results. In our implementation, we use a 3-bit codebook only for $\{r_1, e_1, r_2, e_2\}$ parameters, and 7-bits each for the $\{\, \alpha, \beta \,\}$ multipliers, to make a total of 17 bits. Note that r_1, r_2, are always either 1, 2, or 3, and therefore no more than three rows of unfiltered pixels are needed on the top of any pixel to be filtered.

The principle of subspace projection is illustrated diagrammatically in Fig. 1. Even though none of the cheap restorations X_1, X_2 are close to the source Y, appropriate multipliers

{α, β} can bring them much closer to the source as long as they are moving somewhat in the right direction.

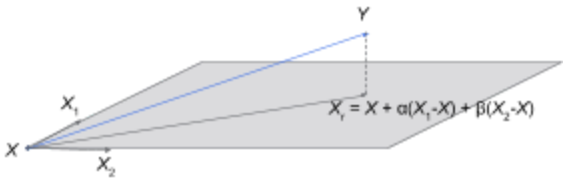

Fig. 1. Subspace projection using cheap restorations to produce a final restoration closer to the source

The principle can be explored further in many different ways. The cheap restorations does not have to be based on self-guided filters. Any two or more cheap restorations could be used.

2.3 Domain Transform Recursive Filters

Domain Transforms [4] are a recent approach to edge-preserving image filtering using only 1-D operations, that can potentially be much faster than other edge-aware processing approaches. We use the recursive filtering incarnation in [4] where the processing steps include horizontal left-to-right and right-to-left recursive order-1 filtering, followed by vertical top-to-bottom and bottom-to-top filtering, conducted over a few (typically 3) iterations. The filter taps are obtained from local horizontal and vertical gradients of the pixels and the iteration index. The specific steps are as follows:

For each iteration $i = 0, 1, \ldots T-1$, for a $M \times N$ image, do:

Left to Right:

$$y[0] = x[0];$$
$$y[n] = (1 - h_{n-1,i})\, x[n] + h_{n-1,i}\, y[n-1], \quad n = 1, 2, \ldots N-1$$

Right to Left:

$$y[N-1] = x[N-1];$$
$$y[n] = (1 - h_{n,i})x[n] + h_{n,i}y[n+1], \quad n = N-2, N-3, \ldots, 0$$

Top to Bottom:

$$y[0] = x[0];$$
$$y[n] = (1 - v_{n-1,i})x[n] + v_{n-1,i}y[n-1], \quad n = 1, 2, \ldots M-1$$

Bottom to Top:

$$y[M-1] = x[M-1];$$
$$y[n] = (1 - v_{n,i})x[n] + v_{n,i}y[n+1], \quad n = M-2, M-3, \ldots, 0$$

where x and y are inputs and filtered outputs respectively at each stage of the operation. The filtering operation is controlled by underlying parameters σ_s, σ_r that have similar interpretation as spatial and range variances of a bilateral filter [1]. The horizontal and vertical taps $h_{n,i}$ and $v_{n,i}$ depend on location n and iteration index i, as follows:,

$$h_{n,i} = a_i^{(d_{x,n}\sigma_s/\sigma_r)}, \quad v_{n,i} = a_i^{(d_{y,n}\sigma_s/\sigma_r)} \qquad (9)$$

where $d_{x,n}$ and $d_{y,n}$ are horizontal and vertical absolute pixel differences at any location n along any row or column, and,

$$a_i = exp(-\sqrt{2}/\sigma_H) \text{ with } \sigma_H = \sigma_s\sqrt{3}(2^{T-i-1})/\sqrt{4^T - 1} \qquad (10)$$

The $d_{x,n}$ and $d_{y,n}$ are determined based on the original pixel values and do not change from iteration to iteration. The $\{\sigma_s, \sigma_r\}$ pair is chosen by the encoder from a small codebook of allowable parameter sets and its index is transmitted to the decoder. The taps $h_{n,i}$ and $v_{n,i}$ for all parameter sets can then be read off from a precomputed 3D lookup table indexed by the parameter set index, iteration index and absolute pixel differences that has 256 possible values for 8-bit content. In our specific implementation we used a 6-bit codebook where σ_s is kept fixed and σ_r takes 64 possible values.

3. SWITCHABLE RESTORATION FRAMEWORK

The above restoration tools are implemented in AV1 in a switchable restoration framework where the tool used changes per *tile*. The tilesize for restoration is selectable betweeen 256 × 256, 128 ×128 or 64 × 64 at the frame level. Typically, for higher resolution sources larger tilesizes are selected. Specifically, for each frame and each component, a *frame_restoration_type* symbol is sent. The values taken by it for the Y-component are as follows:

1. RESTORE_NONE: Do not restore the frame
2. RESTORE_WIENER; Each tile in the frame can either be unrestored or use Wiener restoration
3. RESTORE_SGRPROJ: Each tile in the frame can either be unrestored or use the Self-guided filters w/ Subspace Projection.
4. RESTORE_DOMTXFMRF: Each tile in the frame can either be unrestored or use Domain Transform Recursive Filters.
5. RESTORE_SWITCHABLE: Each tile in the frame can either be unrestored or use any of the other supported tools.

For the chroma components only the first three above are allowed.

Depending on the frame restoration type for each component, for each tile additional information is sent to indicate the actual restoration tool used, and the side-information needed if any. The encoder makes appropriate RD based decisions on what restoration to use. The decoder simply decodes the information sent and applies the filters used.

Note that apart from the domain transform filter, both of the other filters can be implemented on the decoder side with at most 3 rows of unfiltered pixels, available on the top of any pixel. Further the decoder operation is simple enough that the overall tool is quite hardware friendly. Precisely for that reason, since the abstract of this paper was writen, we have removed the Domain Transform filter for consideration in the AV1 codec.

4. CODING RESULTS

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

1. *lowres,* which includes 38 videos of CIF and SIF resolutions.
2. *midres*, which includes 30 videos of 480p and 360p resolution.
3. *hdres*, which contains 38 videos at 720p and 1080p resolution.

150 frames of each video are coded with a single keyframe given a set of target bitrates. The restoration tilesize was fixed at 128 × 128. The domain transform filter is excluded from the results, even though with that included there woud be a modest improvement.

For quality metrics we use average sequence PSNR and SSIM [17] 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:

$$MSE_{combined} = [4MSE_y + MSE_{Cb} + MSE_{Cr}]/6,$$
$$PSNR_{combined} = min\ (\ 10\log_{10}(255^2 / MSE_{combined}),\ 100\ ) \qquad (11)$$

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:

$$SSIM_{combined} = 0.8\ SSIM_y + 0.1\ (SSIM_{Cb} + SSIM_{Cr}) \qquad (12)$$

To compare RD curves obtained by two codecs we use a modified BDRATE [18] metric that uses piecewise cubic Hermite polynomial interpolation (*pchip*) on the rate-distortion points before integrating the difference over a finer grid.

Tables 1-2 below show the BDRATE reduction results for *midres* and *hdres* sets respectively, when the tool that implements loop restoration is turned on using the configuration option *--enable-loop-restoration* over the baseline that includes most other experimental tools provisionally adopted in AV1 as of mid-May 2017, excluding extended transforms and global motion. The **AVERAGE** number at the bottom of each table is the arithmetic average of the BDRATE numbers over all the videos in the same column. BDRATE is computed separately based on the average sequence PSNR and SSIM metrics. The detail lowres results are skipped in the interest of space.

Table 1. AV1 BDRATE results with loop restoration on *midres* set

Video	BDRATE (PSNR)	BDRATE (SSIM)
BQMall_832x480_60.y4m	-0.931	-0.813
BasketballDrillText_832x480_50.y4m	-1.763	0.230
BasketballDrill_832x480_50.y4m	-1.650	-0.167
Flowervase_832x480_30.y4m	-2.637	-3.030
Keiba_832x480_30.y4m	-2.150	-2.021
Mobisode2_832x480_30.y4m	-1.966	-2.040
PartyScene_832x480_50.y4m	-1.673	-0.645
RaceHorses_832x480_30.y4m	-1.987	-1.400
aspen_480p.y4m	-1.697	-1.843
city_4cif_30fps.y4m	-1.900	-1.739
controlled_burn_480p.y4m	-1.731	-0.801
crew_4cif_30fps.y4m	-2.161	-1.835
crowd_run_480p.y4m	-1.250	-1.138
ducks_take_off_480p.y4m	-2.177	-3.153
harbour_4cif_30fps.y4m	-2.332	-1.708
ice_4cif_30fps.y4m	-1.149	-1.068
into_tree_480p.y4m	-1.167	0.114
old_town_cross_480p.y4m	-1.324	-0.939
park_joy_480p.y4m	-0.375	-0.534
red_kayak_480p.y4m	-1.461	-1.438
rush_field_cuts_480p.y4m	-0.734	-0.821
sintel_trailer_2k_480p24.y4m	-2.161	-1.903
snow_mnt_480p.y4m	-0.575	-0.077
soccer_4cif_30fps.y4m	-2.169	-1.206
speed_bag_480p.y4m	-3.109	-5.396
station2_480p25.y4m	-1.302	-0.876
tears_of_steel1_480p.y4m	-1.680	-1.335
tears_of_steel2_480p.y4m	-1.513	-1.164
touchdown_pass_480p.y4m	-1.153	-0.516
west_wind_easy_480p.y4m	-3.320	-2.097
AVERAGE	**-1.707%**	**-1.379%**

Table 2. AV1 BDRATE results with loop restoration on *hdres* set

Video	BDRATE (PSNR)	BDRATE (SSIM)
basketballdrive_1080p50.y4m	-1.594	-0.896
blue_sky_1080p30.y4m	-3.939	-4.320
bqterrace_1080p60.y4m	-2.550	-1.461
cactus_1080p50.y4m	-1.949	-1.480
chinaspeed_xga.y4m	-3.689	-3.386
city_720p30.y4m	-2.287	-1.848
crew_720p30.y4m	-2.575	-2.152
crowd_run_1080p50.y4m	-1.690	-1.137
cyclists_720p30.y4m	-3.233	-2.913
dinner_1080p30.y4m	-3.866	-2.945
ducks_take_off_1080p50.y4m	-3.220	-4.134
factory_1080p30.y4m	-1.043	-0.760
fourpeople_720p60.y4m	-2.570	-3.353
in_to_tree_1080p50.y4m	-1.017	-0.204
jets_720p30.y4m	-2.748	-2.363
johnny_720p60.y4m	-2.093	-2.257
kimono1_1080p24.y4m	-1.710	-1.778
kristenandsara_720p60.y4m	-2.171	-2.443
life_1080p30.y4m	-1.601	-0.287
mobcal_720p50.y4m	-1.005	0.732
night_720p30.y4m	-2.021	-1.712
old_town_cross_720p50.y4m	-1.181	-0.771
parkjoy_1080p50.y4m	-2.163	-2.007
parkrun_720p50.y4m	-1.751	-1.565
parkscene_1080p24.y4m	-0.753	-0.684
ped_1080p25.y4m	-1.533	-1.812
riverbed_1080p25.y4m	-1.622	-1.662
rush_hour_1080p25.y4m	-2.735	-3.889
sheriff_720p30.y4m	-2.250	-2.205
shields_720p50.y4m	-2.391	-1.957
station2_1080p25.y4m	-4.398	-4.337
stockholm_ter_720p60.y4m	-1.583	-1.060
sunflower_720p25.y4m	-5.100	-6.635
tennis_1080p24.y4m	-1.522	-1.557
tractor_1080p25.y4m	-1.893	-1.961
vidyo1_720p60.y4m	-2.973	-3.833
vidyo3_720p60.y4m	-3.358	-5.597
vidyo4_720p60.y4m	-1.935	-2.202
AVERAGE	**-2.308%**	**-2.232%**

Tables 3 shows the overall BDRATE reduction on all three test sets, based on PSNR and SSIM, with our *loop restoration* tool as compared against another tool *CDEF* − also under consideration for the AV1 codec. CDEF is a combination of the deringing filter from Daala [5], and constrained low pass filter [6] from NetVC.

Table 3. Average BDRATE (PSNR and SSIM based) comparisons with alternative configurations.

BDRATE (PSNR)	***lowres***	***midres***	***hdres***
--enable-loop-restoration	**-1.429%**	**-1.707%**	**-2.308%**
--enable-cdef	-0.619%	-0.955%	-1.257%
BDRATE (SSIM)	***lowres***	***midres***	***hdres***
--enable-loop-restoration	**-1.227%**	**-1.379%**	**-2.232%**
--enable-cdef	-0.341%	-0.794%	-1.060%

5. CONCLUSIONS

In this paper we have presented a switchable restoration framework where one of three restoration tools can be selected for each *tile* of a video frame or each frame of a video. Results on varied test sets demonstrate its effectiveness.

6. REFERENCES

[1] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images", *Proc. of the Sixth IEEE International Conference on Computer Vision*, 1998, Bombay, India.

[2] Buades, Antoni, "A non-local algorithm for image denoising". *Proc. IEEE Computer Vision and Pattern Recognition*, 2005. vol. 2: pp. 60–65.

[3] Kaiming He, Jian Sun, and Xiaoou Tang. "Guided image filtering." *Computer Vision–ECCV* 2010. Springer Berlin Heidelberg, 2010. 1-14.

[4] Eduardo S. L. Gastal, Manuel M. Oliveira, "Domain Transform for Edge-Aware Image and Video Processing," *ACM Transactions on Graphics*. Volume 30 (2011), Number 4, Proceedings of SIGGRAPH 2011, Article 69.

[5] J.-M. Valin, A Deringing Filter for Daala... And Beyond, 2016. https://people.xiph.org/~jm/daala/ deringing_demo/

[6] S. Midtskogen, A. Fuldseth, M. Zanaty, "Constrained Low Pass Filter," Network Working Group, Internet Draft, April 2016. https://tools.ietf.org/html/draft-midtskogen-netvc-clpf-02.

[7] Yaniv Romano, John Isidoro, Peyman Milanfar, "RAISR: Rapid and Accurate Image Super Resolution," *IEEE Transactions on Computational Imaging*, vol. PP, Issue 99, 2016.

[8] Mischa Siekmann; Sebastian Bosse; Heiko Schwarz; Thomas Wiegand, "Separable Wiener filter based adaptive in-loop filter for video coding," *Proc. IEEE Picture Coding Symposium (PCS)*, 2010.

[9] M. Naccari, F. Pereira, "Adaptive bilateral filter for improved in-loop filtering in the emerging high efficiency video coding standard," *Proc. Picture Coding Symposium*, May 2012.

[10] Chih-Ming Fu, E. Alshina, A. Alshin, Yu-Wen Huang, Ching-Yeh Chen, Chia-Yang Tsai, Chih-Wei Hsu, Shaw-Min Lei, Jeong-Hoon Park, "Sample Adaptive Offset in the HEVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22(12), December 2012.

[11] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans on Circ. and Sys. for Video Technology*, Vol. 13, No. 7, July 2003.

[12] G. J. Sullivan, J. Ohm, Woo-Jin Han, T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard,". *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 22, Issue: 12, Dec. 2012.

[13] J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins, Y. Xu, *VP8 Data Format and Decoding Guide*, RFC 6386, http://datatracker.ietf.org/doc/rfc6386/

[14] D. Mukherjee, J. Bankoski, R. S. Bultje, A. Grange, J. Han, J. Koleszar, P. Wlkins, 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.

[15] Debargha Mukherjee, Jingning Han, Jim Bankoski, Ronald Bultje, Adrian Grange, John Koleszar, Paul Wilkins, Yaowu Xu, "A Technical Overview of VP9—The Latest Open-Source Video Codec," *Proc. SMPTE Motion Imaging Journal*, Volume: 124, Issue: 1, January/February 2015.

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

[17] 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.

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