Film Grain Synthesis for AV1 Video Codec

Andrey Norkin[*] and Neil Birkbeck[+]

[*]*Netflix*
100 Winchester Cir
Los Gatos, CA 95032, USA
anorkin@netflix.com

[+]*Google*
1600 Amphitheatre Pkwy
Mountain View, CA, USA
birkbeck@google.com

***Abstract**:* Film grain is abundant in TV and movie content. It is often part of the creative intent and needs to be preserved while encoding. However, the random nature of film grain is difficult to compress using traditional coding tools. This paper describes a film grain modeling and synthesis algorithm proposed for the AV1 video codec. At the encoder, an autoregressive model of film grain is transmitted relative to a denoised signal, and the film grain strength is modeled as a function of intensity. The corresponding renoising at the decoder is implemented using an efficient block-based approach suitable for use in consumer electronic devices. Preliminary results indicate that the approach can give significant bitrate savings (up to 50%) on sequences with heavy film grain.

1. Introduction

In the entertainment industry, film grain is widely present in the motion picture and TV material and is considered part of the creative intent. The grain is inherent in analog motion picture film due to the process of exposure and development of silver-halide crystals dispersed in photographic emulsion [2] as randomly distributed grains appear at the locations where the silver crystals have formed. Digital cameras do not produce film grain; however, in post-production, film grain is often added to the captured material to create the "movie" look. Therefore, when encoding motion picture and TV content, it is important to preserve film grain to maintain the creative intent of the content creators.

Another type of noise in videos is created by digital camera sensors. This noise is usually uncorrelated between the samples and has different characteristics from film grain. However, after scaling and compression (e.g., at the mobile device) the video can produce noise patterns similar to the film grain.

Film grain has the following characteristics: 1) the noise is spatially correlated and grains can be more than one sample in size, 2) noise is temporally independent, and 3) originally, film grain in color components is independent and follow a multiplicative noise model. However, the latter was found to not be true for significant part of content with film grain in the Netflix database. The conceived reasons are color component transformation followed by downsampling of chroma components and various post-processing applied to the video at the post-production stage.

Since film grain and sensor noise shows a high degree of randomness, it is difficult to compress efficiently. Randomness makes prediction difficult, motion estimation less precise, and the prediction residual in case of motion compensation contains noise with twice the variance of the film grain. Prohibitively high bitrates are often required to reconstruct film grain with sufficient quality. Moreover, from our experience, some encoding tools, such as a de-ringing filter, may additionally suppress film grain.

There have been a number of attempts to address the film grain encoding problem. A typical approach involved detecting the film grain, denoising the source video and estimating the film grain parameters. The film grain can later be synthesized from the parameters estimated at the encoder side and added to the reconstructed video. Some approaches to this problem have been previously described in the literature [1], [2], [3], [4], and [6]. For film grain modeling, [2], [4], and [6] used autoregressive (AR) process. A coded film grain sample with mirroring and rotation transformations was used in [3] to synthesize the grain in the decoder. A number of approaches addressed film grain noise removal, ranging from using a video encoder as denoiser [3] to a Wiener type filter [1], non-linear denoising with total variation minimization [6] and multi-hypothesis motion compensated filtering [4], [5]. Film grain parameters are typically estimated on flat regions of the picture. An optional film grain characteristics supplemental enhancement information (SEI) message was defined in the H.264 video compression standard [8]. The technology was mostly based on the methods proposed in [2], which comprised generation of film grain based on filtering of the Gaussian noise in the transform domain and an AR process. An additive and multiplicative noise models were supported along with specifying intesity intervals that could use different film grain. The same SEI message is supported in the HEVC standard [9]. These SEI messages, however, are not frequently supported in video decoders, which makes it difficult to rely on their presence in an open video distribution system.

This paper presents a film grain syntesis tool proposed for the AV1 video codec [10] in the Alliance for Open Media (AOM). The motivation for including this tool is to provide mandatory support of the film grain synthesis in the AV1 video codec. The proposed tool uses an autoregressive model to support a range of different noise characteristics, varying from film grain to sensor noise and compressed sensor noise (Section 3). Additionally, the tool also supports flexible modeling of the relationship between film grain strength and signal intensity, which includes both additive and multiplicative models (Section 4). The noise parameters are signaled in the bitstream and an efficient block-based approach is proposed to synthesize the noise at decoder (Section 5). Preliminary subjective comparison indicates that the proposed approach can give significant bitrate savings (up to 50%) on sequences with heavy film grain noise (Section 6).

2. Proposed film grain framework

The proposed film grain modeling and synthesis framework is shown in Fig. 1. The film grain is removed from the video by a denoising process, and the grain parameters are estimated from the flat regions of the difference between the noisy and de-noised versions of the video sequence; these parameters are sent along with the compressed video bitstream.

After the decoding, the film grain is synthesized and added to the reconstructed video frames. A number of approaches have been proposed in the literature for film grain denoising, including those described in [1], [4], [5], and [6]. Although high quality film grain removal is an important step in the overall system design, this paper does not address this topic in great detail and focuses on aspects of film grain modeling and synthesis.

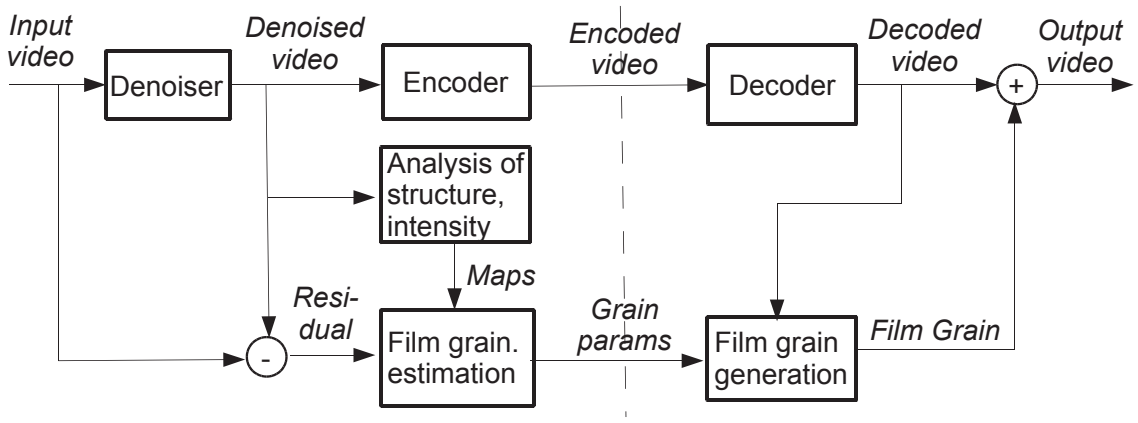


Figure 1: Proposed framework for film grain estimation and synthesis

Figure 2: Areas used for film grain estimation (black)

The details of the algorithm are as follows. When the film grain/noise parameters are estimated, it is important to make sure that only smooth regions of the picture are used in estimation, since edges and textures can affect estimation of the film grain strength and pattern. To determine smooth areas of the picture, the Canny edge detector [7] is applied to the denoised image at different scales, followed by the dilation operation. The thresholds in Canny detector are proportional to the local intensity of the signal and are therefore adapted to the luminance. Figure 2 shows an example of the image that is produced by applying Canny edge detection at three scales to the denoised video followed by dilation. The low luminance areas are also excluded from film grain pattern estimation.

After the regions to detect the film grain are determined, film grain intensity and pattern are determined in these areas, as described in the following section.

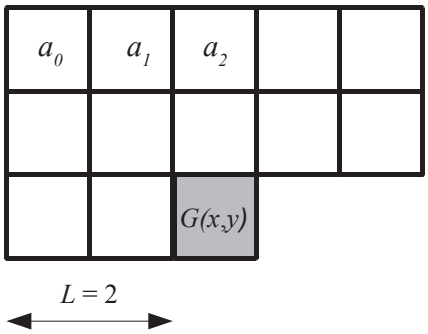
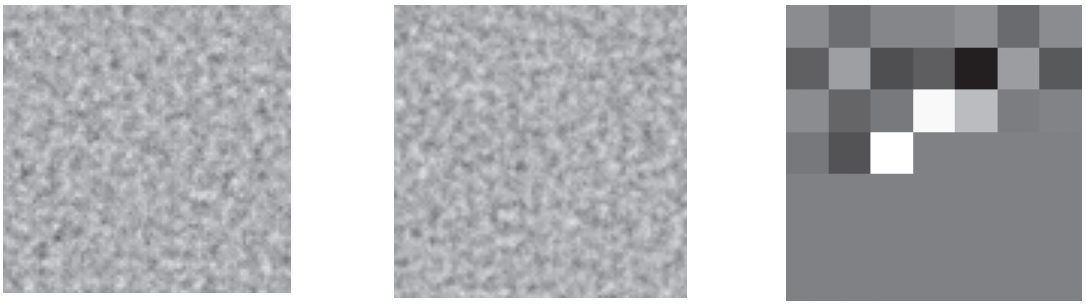


Figure 3: Current sample of film grain $G(x, y)$ with AR coefficients



(a) Original grain (b) Synthesized grain (c) AR coefficients

Figure 4: Example of film grain synthesis and corresponding AR coefficients

3. Modeling film grain pattern

The film grain pattern is modeled with an autoregressive process. This autoregressive model is used to synthesize a template with film grain at the decoder.

Let $G(x, y)$ be a zero-mean film grain sample at the current position. For lag $L = 2$, the grain sample G is calculated as follows (see Fig. 3):

$$G(x, y) = a_0\, G(x - 2, y - 2) + a_1\, G(x - 1, y - 2) + a_2\, G(x, y - 2) + \ldots + z, \qquad (1)$$

where $a_0 \ldots a_n$ are AR-coefficients, $G(x + k, y + m)$ are previous film grain sample values in the causal neighborhood, and z is the unit-variance Gaussian noise. The Gaussian variable z can be obtained from a predefined set stored at the decoder and encoder side. The number of AR-coefficients a_i is determined by the lag parameter L and is equal to $2L(L + 1)$ for luma and $2L(L + 1) + 1$ for chroma component. In chroma components, there is one additional coefficient a_n to capture correlation with a luma grain sample at the same spatial position. The value of AR-coefficients lag L can take values from 0 to 3. In this case, $L = 0$ corresponds to the case of modeling Gaussian noise whereas higher values of L may correspond to film grain with larger size of grains. The AR-coefficients $a_0 \ldots a_n$ are estimated by a method based on the Yule-Walker AR equations. An example of film grain synthesis and visualization of AR coefficients is shown in Fig. 4. Brighter colors in Fig. 4(c) correspond to higher values, and the background gray to zero.

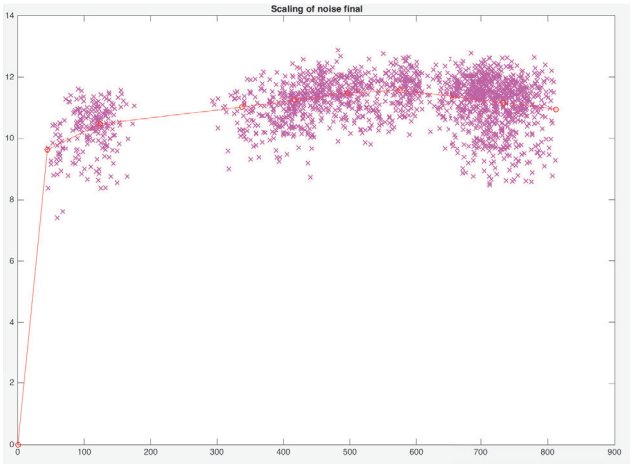


Figure 5: Scaling of film grain in luma, represented with a piece-wise linear function. Horizontal axis corresponds to luma values, vertical to standard deviation of film grain.

4. Modeling film grain intensity

Film grain strength can vary with signal intensity. As mentioned previously, film grain in the final material is not necessarily well represented with a multiplicative model. The proposed approach uses direct coding of the film grain strength σ with a piece-wise linear function for each of Y, Cb, and Cr color components. When adding film grain to the luma component, the following model is used:

$$\mathrm{Y}' = \mathrm{Y} + f(\mathrm{Y})\, G_L, \tag{2}$$

where Y′ is the resulting luma re-noised with film grain, Y is the reconstructed value of luma (before adding film grain), and G_L is the luma film grain sample. Here, f (Y) is a piece-wise linear function that scales film grain depending on the luma component value that is fit by measuring noise strength on flat regions of the input. This piece-wise linear function can be implemented as a precomputed look-up table (LUT) that is initialized before running the grain synthesis. The LUT (for both 8-bit or 10-bit video) have 256 entries with 8-bit values. For video with bit depth higher than 8, linear interpolation is used to obtain values between the LUT entries. Fitting the scaling function to the data can be done with various methods. In the simulations for this paper, the scaling function was determined by using least squares fit to the local standard deviations of the flat areas to their local mean intensity values. Some additional criteria have been used, such as that scaling functions is equal to zero for the zero luma values. Figure 5 shows an example of representing the standard deviation of the film grain with a piece-wise linear function.

Scaling film grain for the chroma component is done as follows. For a chroma component (e.g. Cb), the noise is modulated using the following formula:

$$\mathrm{Cb}' = \mathrm{Cb} + f(u)\, G_{\mathrm{Cb}}, \tag{3}$$

$$u = b_{\mathrm{Cb}}\, \mathrm{Cb} + d_{\mathrm{Cb}}\, \mathrm{Y}_{\mathrm{av}} + h, \tag{4}$$

where u is the index into the look-up table that corresponds to a Cb component scaling function. The index depends on both the Cb and luma components for the current pixel, which reflects the fact that the film grain in chroma may depend on the luma component

(e. g. film grain in chroma may be close to zero in very low luminance signal and significant in the gray and white areas, although chroma values may be similar in both cases). The value of Y_{av} is the average value of luma corresponding to this chroma sample (taken from one line of samples). For example, in 4:2:0 YCbCr, $Y_{av} = (Y_1 + Y_2 + 1) >> 1$, where Y_1, Y_2 are neighboring (co-located) luma samples on the even line.

5. Film grain synthesis algorithm

To reconstruct the film grain at the decoder side, the encoder/pre-processing module sends the following parameters to the decoder: lag value L, AR-coefficients $a_0 \ldots a_n$, a set of points for a piece-wise linear scaling function for each color component, and values b_{Cb}, d_{Cb}, h_{Cb} , b_{Cr}, d_{Cr}, h_{Cr} for LUT index of chroma components. All values are signaled as (quantized) integers and take insignificant bandwidth compared to the coded video, especially at higher resolutions. The film grain parameters are signaled for the highest (display) resolution of the reconstructed video frames (AV1 [10] supports varying frame resolution, which allows coding some frames at a lower resolution than the display resolution). The parameters can take up to 145 bytes (including 64 bytes for scaling functions and 74 bytes for AR coefficients). The parameters are signaled once until later updated or for every frame if necessary. Sending parameters for every frame may help to maintain temporal consistency when film grain characteristics are changing gradually.

5.1. Film grain synthesis and re-noising

With received AR coefficients $a_0 \ldots a_n$ and stored Gaussian noise values, an AR process is run in a raster scan order to generate a 64×64 luma film grain template (and two 32×32 chroma templates). The padding of $L + 3$ for chroma and L + 6 for luma is used when generating a template. Then, the film grain is consecutively applied on a 32×32 luma block basis to the reconstructed picture in the raster scan order. Luma grain blocks of size 32×32 are randomly selected from the 64×64 template, the grain samples are scaled with the scale function LUT, described in Section 4 and added to the reconstructed sample values, followed by clipping. The 16×16 chroma blocks are collocated with corresponding luma blocks in the template. Selection of the 32×32 film grain block from the template is shown in Fig. 6.

The pseudo-random generator is a shift-back linear-feedback shift register (LFSR) based on XOR of the length of 16 bits. The XOR-ed values 16, 14, 13, and 11 correspond to the feedback polynomial $x^{16} + x^{15} + x^{13} + x^4 + 1$. The offsets s_x and s_y are generated using four and next four most significant bits on the register. The chroma offsets range from 0 to 15, while luma offsets are in multiples of two of chroma offsets. To enable parallel processing, the generator is initialized in the beginning of each row of 32×32 blocks with an expression that depends on the sum of syntax elements and the row number.

The operation for adding grain to the samples of a 32×32 luma block is as follows:

$$Y'(x, y) = \text{clip3}(Y(x, y) + ((G_L(x + s_x, y + s_y) * f(Y) + 2^{\text{shift}-1}) >> \text{shift}), a, b), \quad (5)$$

where a and b define the legal range, x and y are coordinates inside the block, and parameter shift controls scaling of the film grain.

When the template contains grain with relatively low frequencies, applying film grain in 32×32 patches can result in visible block artifacts. To mitigate these effects, the algorithm has an option of overlap between the noise blocks. In this case, the luma block

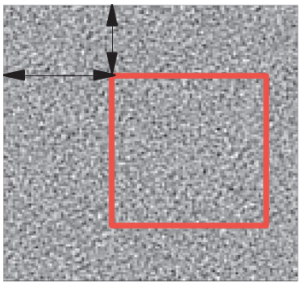


Figure 6: Film grain block (32×32) taken from 64×64 template with random *x* and *y* offsets

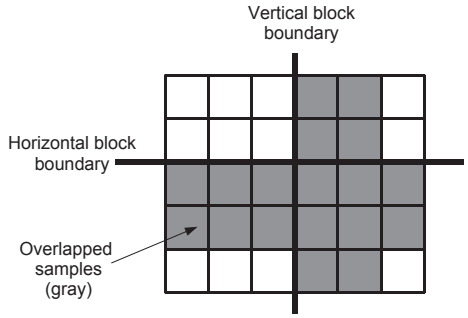


Figure 7: Overlapped blocks: current block samples overlap only with the blocks to the right and below and not with the previously processed blocks.

size is 34×34 with the two last rows and two last columns overlapping with the blocks to the right and to the bottom. The size of chroma blocks is then 16×16 with 1 sample overlap. The example of the overlapped luma blocks is shown in Fig. 7. The blocks overlap happens from the current block to the blocks below and to the right but not in the opposite direction. Therefore, no sample line buffer is needed.

The overlap operation is applied to the noise samples before adding them to the reconstructed picture. When the block-wise processing is used, the decoder may need to store template offsets s_x and s_y for the grain blocks in the previous row in order to extract the overlapping rows of samples. The operation used for overlapped film grain samples over horizontal block boundaries in luma is as follows:

$$G_{cur}(x, 0) = (27 * G_{up}(x, 32) + 17 * G_{cur}(x, 0) + 16) >> 5, \quad (6)$$

$$G_{cur}(x, 1) = (17 * G_{up}(x, 33) + 27 * G_{cur}(x, 1) + 16) >> 5, \quad (7)$$

where $G_{cur}(x, 0)$ are samples of row 0 of the current block and $G_{up}(x, 32)$ denotes samples of row 32 of the upper block. A similar (transposed) operation is used for grain blocks overlap across vertical block boundaries. Overlap between vertical block boundaries is applied first, followed by overlap between horizontal boundaries. The overlap operation between chroma blocks is done as follows:

$$G_{cur}(x, 0) = (23* G_{top}(x, 16) + 22 * G_{cur}(x, 0) + 16) >> 5 \quad (8)$$

One can notice that the sum of the weights corresponding to the contribution of the blocks to the overlapped samples is not equal to one. This result is intended since to keep the constant variance of the noise, the sum of squares of the coefficients should add up to one (provided that the samples are uncorrelated, which approximately holds for boundary samples of the blocks obtained by random s_x and s_y offsets inside the template).

Generally, using overlapped film grain windows instead of deblocking helps preserve high frequencies, which may result in higher subjective quality.

5.2. Memory requirements

The proposed algorithm generally requires additional memory for the re-noised pictures. The algorithm should not be applied inside the video decoder loop since the reconstructed

pictures used as references should be kept noise-free to provide bitrate savings. However, the algorithm can be applied as part of the display processing chain.

When it comes to hardware implementation, the algorithm may require storing noise templates (one for luma and two for chroma) in on-chip memory. The samples of the template are stored with depth of 8 bits, which results in memory requirements of 4096 bytes for the luma template and 1024 bytes for each of the chroma templates for 8-bit video. Please note that if chroma components are processed independently from luma, on-chip memory requirements are not additive and determined by the luma template size.

When the grain block overlap is used, the template offsets for the previous row of 32×32 blocks need to be stored. The offsets s_x and s_y each require 4-bits. Therefore, memory required for storing the offsets is 60 bytes for 1080p resolution and 120 bytes for 3840×2160p resolution.

6. Results

The algorithm performance has been evaluated based on several video sequences with heavy film grain. Since the proposed technology removes film grain and adds similar grain, the objective metrics would not work well for this comparison. Therefore, an informal subjective comparison was used to evaluate the results of the algorithm.

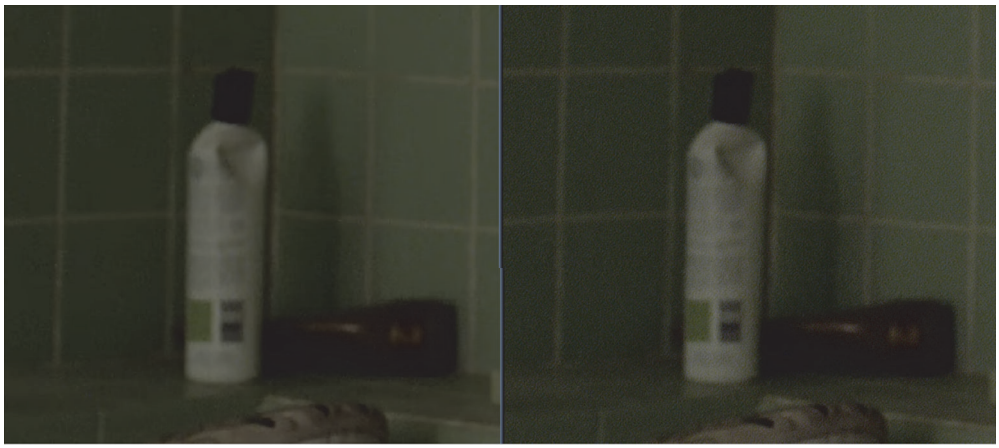
The video sequences were compressed using the AV1 video codec [11]. To test the proposed algorithm, the film grain was first removed from the sequences by hqdn3d denoising [12] from the ffmpeg package with parameters 6:6:6:6. The denoised sequences were compressed with the AV1 codec. Film grain was then added to the reconstructed sequences using the proposed algorithm. Note that the hqdn3d is a somewhat simple 3D denoising algorithm, not adapted for removing film grain. The subjective results would likely improve if a film-grain adaptive denoising is used. The following parameters were used for configuring the AV1 codec [11]:

```
--input-bit-depth=10 --end-usage=q --cq-level=30 --lag-in-frames=25 --
auto-alt-ref=2 --profile=2 --bit-depth=10 --cpu-usage=2
```

Figure 8 illustrates the subjective performance of the algorithm. The examples are cropped parts of the following video sequences: BreakingBad (a-b), 3840×2160@24fps, and TaxiDriver (c-d) and LawrenceOfArabia (e-f), 1920×1080@24fps. The first two sequences were encoded at QP30, and the latter at QP25. One can notice that the pictures on the right obtained with the proposed algorithm have better preserved film grain despite lower bitrate than those on the left where film grain is directly encoded. The subjective look of the motion video obtained with the proposed approach is also better and the film grain pattern is more temporally stable than the grain encoded by the AV1 directly, in which case grain is often completely removed or it manifests pulsing because of temporal quantizer variation.

7. Conclusions and discussion

The paper has presented film grain modeling and synthesis tool proposed for the AV1 video codec [10]. The tool helps to preserve a film grain look of the encoded video while keeping significantly lower bitrate compared to the scenario when the film grain is directly encoded. The proposed model can support a range of different film noise



(a) Coded at 22 137 kbps (b) With noise synthesis at 5 833 kbps

(c) Coded at 13 112 kbps (d) With noise synthesis at 5 799 kbps

(e) Coded at 5 729 kbps (f) With noise synthesis at 2 821 kbps

Figure 8: Parts of the reconstructed frames; (a), (c), (e): AV1, (b), (d), (f): AV1 with proposed algorithm

characteristics, varying from film grain to sensor noise. The most complex parts of the algorithm are at the encoder side, while the synthesis at the decoder side is rather inexpensive, except the need to write the renoised pictures to the memory, which can be mitigated by implementing the algorithm as part of the display processing chain. As such, the algorithm is suitable for implementing in the consumer electronics devices. Testing the algorithm on content with heavy film grain indicates that the subjective quality of the encoded content can be improved while the bitrate decreased.

The results presented in this paper used a rather simple algorithm for denoising the input video. The authors believe that the results for both encoding and noise parameters estimation can be improved if better denoising algorithms for film grain are used, which could be a part of further research.

8. Acknowledgements

The authors would like to thank Ioannis Katsavounidis, Anil Kokaram, Cristina Rosu, Mohammad Izadi, Balu Adsumilli, Minhua Zhou, Ranga Srinivasan, Samuel Wong, Eric Chai and a number of other video compression experts in AOM community for valuable advice on algorithm details and attention to its implementation complexity.

References

[1] J. C. Kit Yan and D. Hatzinakos, "Signal-dependent film grain noise removal and generation based on higher-order statistics", in *Proc. IEEE Signal Processing Workshop on Higher-Order Statistics*, July 1997, Banff, Canada.

[2] C. Gomila, A. Kobilansky, “SEI message for film grain encoding”, ISO/IEC JTC1/SC29/WG11, ITU-T SG16 Q.6 document JVT-H022, Geneva, CH, May 2003.

[3] M. Schlockermann, S. Wittman, T. Wedi, “Film grain coding in H.264/AVC”, ISO/IEC JTC1/SC29/WG11, ITU-T SG16 Q6 doc. JVT-I034, Sep. 2003, San Diego, CA

[4] J. Dai, O. C. Au, C. Pang, W. Yang, F. Zou, "Film grain noise removal and synthesis in video coding", in *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing (ICASSP)*, Mar. 2010, Dallas, TX, USA

[5] I. Hwang, J. Jeong, J. Choi, Y. Choe, "Enhanced Film Grain Noise Removal for High Fidelity Video Coding", *in IEEE Int. Conf. on Information Science and Cloud Computing Companion (ISCC-C)*, Dec. 2013, Guangzhou, China.

[6] B. T. Oh, S. Lei, C.-C. J. Kuo, "Advanced Film Grain Noise Extraction and Synthesis for High-Definition Video Coding", in *IEEE Trans. on Circ. and Systems for Video Technol.*, Vol. 19, Dec. 2009.

[7] J. Canny, “A Computational Approach To Edge Detection”, in *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6): 679–698, 1986.

[8] Recommendation ITU-T H.264 (04/2017): "Advanced video coding for generic audiovisual services".

[9] Recommendation ITU-T H.265 (12/16): "High efficiency video coding".

[10] AOMedia AV1 Specification Document, https://aomedia.googlesource.com/av1-spec/

[11] AV1 Codec source code repository, https://aomedia.googlesource.com/aom

[12] hqdn3d denoising, http://avisynth.nl/index.php/Hqdn3d