# The Daala Directional Deringing Filter

Jean-Marc Valin, *Member, IEEE*

***Abstract*—This paper presents the deringing filter used in the Daala royalty-free video codec. The filter is based on a non-linear conditional replacement filter and is designed for vectorization efficiency. It takes into account the direction of edges and patterns being filtered. The filter works by identifying the direction of each block and then adaptively filtering along the identified direction. In a second pass, the blocks are also filtered in a different direction, with more conservative thresholds to avoid blurring edges. The proposed deringing filter is shown to improve the quality of both Daala and the Alliance for Open Media (AOM) AV1 video codec.**

## I. Introduction

The Daala video codec has been under development since 2012 with the goal of achieving royalty-free status by using technology that differs greatly from more traditional video codecs such as H.264 and HEVC. For example, Daala uses lapped transforms [1], [2] rather than relying on more traditional deblocking filters. It is also based on the perceptual vector quantization(PVQ) [3] technique rather than on scalar quantization of a transformed residual. Some of these techniques achieve better quality on textured content, at the cost of more ringing artefacts. These are especially present in sharp edges due to Daala's lack of intra prediction modes capable of predicting diagonal directions. For these reasons, Daala greatly benefits from a deringing filter.

The main goal of deringing is to filter out ringing artifacts while retaining all the details of the image. In HEVC, this is achieved by the Sample Adaptive Offset (SAO) [4] algorithm that defines signal offsets for different classes of pixels. Unlike SAO, the approach we take in Daala is that of a non-linear spatial filter. From the very beginning, the design of the filter was constrained to be easily vectorizable (i.e. implementable with SIMD operations), which was not the case for other non-linear filters like the median filter [5] and the bilateral filter [6].

The design of the deringing filter originates from the following observations. The amount of ringing in a coded image tends to be roughly proportional to the quantization step size. The amount of detail is a property of the input image, but the smallest detail actually retained in the quantized image tends to also be proportional to the quantization step size. For a given quantization step size, the amplitude of the ringing is generally less than the amplitude of the details.

This paper describes a deringing filter that takes into account the direction of edges and patterns being filtered.

Jean-Marc Valin is with Mozilla Corporation and Xiph.Org Foundation. Corresponding email: jmvalin@jmvalin.ca



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 |
| 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 |
| 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 |
| 3 | 3 | 4 | 4 | 5 | 5 | 6 | 6 |
| 4 | 4 | 5 | 5 | 6 | 6 | 7 | 7 |
| 5 | 5 | 6 | 6 | 7 | 7 | 8 | 8 |
| 6 | 6 | 7 | 7 | 8 | 8 | 9 | 9 |
| 7 | 7 | 8 | 8 | 9 | 9 | 10 | 10 |

Figure 1. Line number $k$ for pixels following direction $d = 1$ in an $8 \times 8$ block.

The filter works by identifying the direction of each block and then adaptively filtering along the identified direction. In a second pass, the blocks are also filtered in a different direction, with more conservative thresholds to avoid blurring edges. The deringing filter also vectorizes well, requiring no per-pixel scalar operation. An high-level interactive demonstration of the algorithm is available at [7].

## II. Direction Search

The proposed deringing filter is based on the direction of edges, so we will start by describing the direction search. For this, the image is first divided into blocks of $8 \times 8$. The block size is chosen to be fine enough to adequately handle non-straight edges, while being large enough to reliably estimate directions when applied to a quantized image. Having a constant direction over an $8 \times 8$ region also makes vectorization of the filter easier.

For each block we want to determine the direction that best matches the pattern in the block. This is done by minimizing the sum of squared differences (SSD) between the quantized block and a perfectly directional block. A perfectly directional block is a block for which each line along a certain direction has a constant value. For each direction, we assign a line number $k$ to each pixel, as shown in Fig. 1.

For each direction $d$, the pixel average for line $k$ is determined by:

$$\mu_{d,k} = \frac{1}{N_{d,k}} \sum_{p \in P_{d,k}} x_p \ , \tag{1}$$

where $x_p$ is the value of pixel $p$, $P_{d,k}$ is the set of pixels in line $k$ following direction $d$ and $N_{d,k}$ is the cardinality of

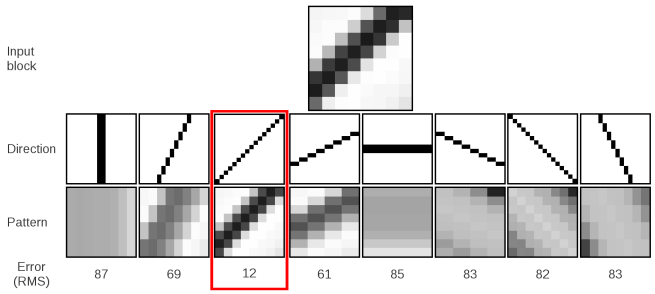


Figure 2. Example of direction search for an $8 \times 8$ block. The patterns shown are based on the $\mu_{d,k}$ values. In this case, the 45-degree direction is the one that minimizes $\sigma_d^2$, so it would be selected by the search. Note that the error values $\sigma_d$ shown are never computed in practice (only $s_d$ is).

$P_{d,k}$ (for example, in Fig. 1, $N_{1,0} = 2$ and $N_{1,4} = 8$). The SSD is then defined as:

$$\sigma_d^2 = \sum_{k \in \text{block},d} \left[ \sum_{p \in P_{d,k}} \left( x_p - \mu_{d,k} \right)^2 \right] \quad . \tag{2}$$

Expanding (2) and then substituting (1) into it, we get

$$\begin{aligned} \sigma_d^2 &= \sum_{k \in \text{block},d} \left[ \sum_{p \in P_{d,k}} x_p^2 - 2 \sum_{p \in P_{d,k}} x_p \mu_{d,k} + \sum_{p \in P_{d,k}} \mu_{d,k}^2 \right] \\ &= \sum_{k \in \text{block},d} \left[ \sum_{p \in P_{d,k}} x_p^2 - 2\mu_{d,k} \sum_{p \in P_{d,k}} x_p + N_{d,k} \mu_{d,k}^2 \right] \\ &= \sum_{k \in \text{block},d} \left[ \sum_{p \in P_{d,k}} x_p^2 - \frac{1}{N_{d,k}} \left( \sum_{p \in P_{d,k}} x_p \right)^2 \right] \\ &= \sum_{p \in \text{block}} x_p^2 - \sum_{k \in \text{block},d} \frac{1}{N_{d,k}} \left( \sum_{p \in P_{d,k}} x_p \right)^2 \quad . \end{aligned} \tag{3}$$

Note that the simplifications leading to (3) are the same as to those allowing a variance to be computed as $\sigma_x^2 = \sum x^2 - \left( \sum x \right)^2 / N$. Considering that the first term of (3) is constant with respect to $d$, we simply find the optimal direction $d_{opt}$ by maximizing the second term:

$$d_{opt} = \max_d s_d \, , \tag{4}$$

where

$$s_d = \sum_{k \in \text{block},d} \frac{1}{N_{d,k}} \left( \sum_{p \in P_{d,k}} x_p \right)^2 \quad . \tag{5}$$

We can avoid the division in (5) by multiplying $s_d$ by 840, the least common multiple of the possible $N_{d,k}$ values ($1 \leq N_{d,k} \leq 8$). When using 8-bit pixel data (for higher bit depths, we downscale to 8 bits), and centering the values such that $-128 \leq x_p \leq 127$, then $840 s_d$ and all calculations leading to that value fit in a 32-bit signed integer.

Fig. 2 shows an example of a direction search for an $8 \times 8$ block containing a line.

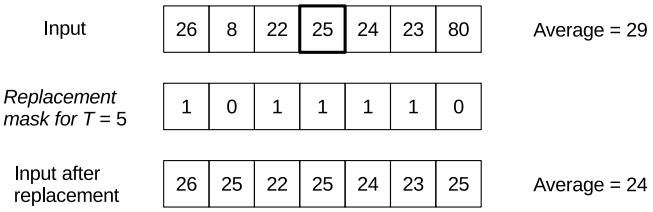


Figure 3. Conditional replacement filter computation

## III. Conditional Replacement Filter

The conditional replacement filter is designed to remove noise without blurring sharp edges. A low-pass finite impulse response (FIR) filter with $(2M + 1)$ taps in one dimension can be expressed as

$$y(n) = \frac{1}{W} \sum_{k=-M}^{k=M} w_k x(n+k) \quad , \tag{6}$$

where $W = \sum_{k=-M}^{M} w_k$. Although it removes noise from a signal, it also blurs any details and sharp edges, which is undesirable. Bilateral filters avoid the blurring effect by making each weights $w_k$ dependent on the difference signal $x(n+k) - x(n)$, typically using a Gaussian function. One disadvantage of this approach is that it requires keeping track of the sum of the weights for each sample $n$ being filtered. It also results in having a different $\frac{1}{W}$ normalization factor for each pixel, making vectorization harder.

Rather than change the $w_k$ values, the conditional replacement filter changes the signal $x(n+k)$ used in the filter. When filtering sample $n$, any of the $x(n+k)$ tap inputs that differs from $x(n)$ by more than a threshold $T$, is replaced by $x(n)$ in the calculation:

$$y(n) = \frac{1}{W} \sum_{k=-M}^{k=M} w_k R(x(n), x(n+k), T) \quad , \tag{7}$$

with

$$R(x_0, x_k, T) = \begin{cases} x_k & , |x_k - x_0| < T \\ x_0 & , \text{otherwise} \end{cases} \quad . \tag{8}$$

The filter computation is illustrated in Fig. 3 and an example is shown in Fig. 4.

Through algebraic simplifications, we can express the filter (7) in terms of the differences $x(n+k) - x(n)$, which yields

$$y(n) = x(n) + \frac{1}{W} \sum_{k=-M, k \neq 0}^{k=M} w_k f(x(n+k) - x(n), T) \quad , \tag{9}$$

with the threshold function

$$f(d, T) = \begin{cases} d & , |d| < T \\ 0 & , \text{otherwise} \end{cases} \quad . \tag{10}$$

The advantage of this formulation is that the normalization by $\frac{1}{W}$ can be approximated without causing any bias, even when $W$ is not a power of two. Also, because $W$ does

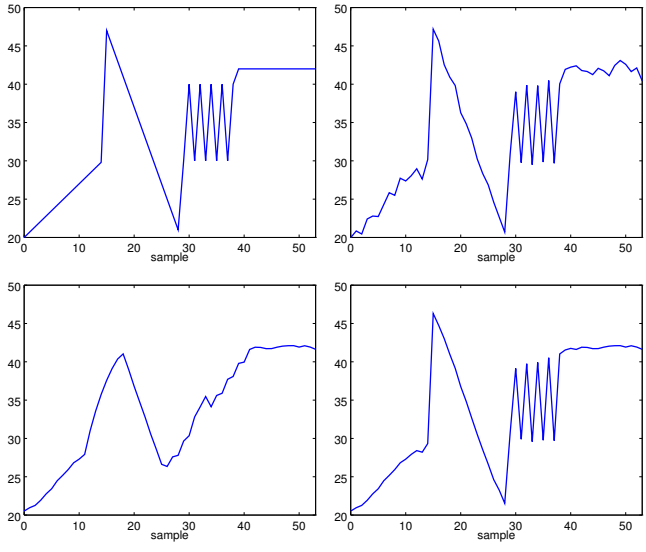


Figure 4. Conditional replacement filter example. Up-left: original signal, up-right: noisy signal, bottom-left: filtered with 7-tap linear filter, bottom-right: filtered with 7-tap conditional replacement filter.

Table I
DIRECTION PARAMETERS

| Index | Direction | $d_x$ | $d_y$ |
|---|---|---|---|
| 0 | ↗ | 1 | $-1$ |
| 1 | ↗ | 1 | $-\frac{1}{2}$ |
| 2 | ↔ | 1 | 0 |
| 3 | ↘ | 1 | $\frac{1}{2}$ |
| 4 | ↘ | 1 | 1 |
| 5 | ↘ | $\frac{1}{2}$ | 1 |
| 6 | ↕ | 0 | 1 |
| 7 | ↙ | $-\frac{1}{2}$ | 1 |

not depend on the number of pixels being replaced, the normalization is easy to vectorize over a row (or column) of pixels.

### A. Directional Filtering

The directional filter for pixel $(i, j)$ is defined as the 7-tap conditional replacement filter

$$y(i,j) = x(i,j) + \frac{1}{W} \sum_{k=-3,\, k\neq 0}^{3} w_k f\left[x(i,j) - x\left(i + \lfloor kd_y \rfloor, j + \lfloor kd_x \rfloor\right), T_d\right] \quad (11)$$

where $d_x$ and $d_y$ define the direction, $W$ is a constant normalizing factor, $T_d$ is the filtering threshold for the block. The direction parameters are shown in Table I. The weights $w_k$ can be chosen so that $W$ is a power of two. For example, Daala currently uses $\mathbf{w} = \begin{bmatrix} 1 & 2 & 3 & (4) & 3 & 2 & 1 \end{bmatrix}$ (where the middle value of 4 is implicit) with $W = 16$. Since the direction is constant over $8 \times 8$ blocks, all operations in this filter are directly vectorizable over the blocks.

### B. Second Stage Filter

The 7-tap directional filter is sometimes not enough to eliminate all ringing, so we use an additional filtering step

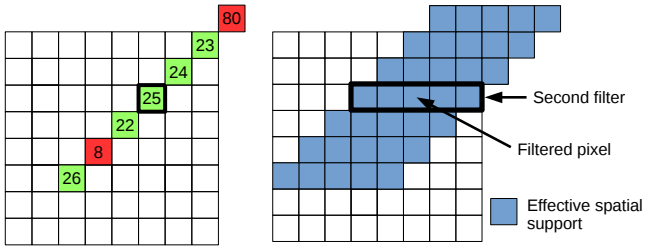


Figure 5. Filtering along the direction (left) and filtering across the direction (right).

Table II
SECOND STAGE FILTER PARAMETERS

| Index | Direction | $d_x$ | $d_y$ |
|---|---|---|---|
| 0 | ↔ | 1 | 0 |
| 1 | ↕ | 0 | 1 |
| 2 | ↕ | 0 | 1 |
| 3 | ↕ | 0 | 1 |
| 4 | ↔ | 1 | 0 |
| 5 | ↔ | 1 | 0 |
| 6 | ↔ | 1 | 0 |
| 7 | ↔ | 1 | 0 |

that operates across the direction lines used in the first filter. Considering that the input of the second filter has considerably less ringing than the input of the second filter, and the fact that the second filter risks blurring edges, the position-dependent threshold $T_2(i, j)$ for the second filter is set lower than that of the first filter $T_d$. The filter structure is the same as the one in Eq. (11). The direction parameters for the second stage filter are shown in Table (II) and the filter weights are $\mathbf{w} = \begin{bmatrix} 1 & 1 & (4/3) & 1 & 1 \end{bmatrix}$ (the middle value $4/3$ is again implicit) with $W = 16/3$. Considering that each input pixel from the second filter is itself the output of the 7-tap directional filter, the combination of the two filters is effectively a 35-tap separable filter.

## IV. SETTING THRESHOLDS

The thresholds $T_d$ and $T_2$ must be set high enough to smooth out ringing artifacts, but low enough to avoid blurring important details in the image. Although the ringing is *roughly* proportional to the quantization step size $Q$, as the quantizer increases the error grows slightly less than linearly because the unquantized coefficients become very small compared to $Q$. As a starting point for determining the thresholds, Daala uses a power model of the form

$$T_0 = \alpha_1 Q^{\beta} \ell \ , \quad (12)$$

with $\beta = 0.842$ in Daala, and where $\alpha_1$ depends on the input scaling. The deringing *level* $\ell$ is a threshold adjustment coded for each superblock ($64 \times 64$). In the AV1 codec, a global threshold is selected by the encoder instead of using a function of the quantizer, so

$$T_0 = \ell_g \cdot \ell \ . \quad (13)$$

Another factor that affects the optimal filtering threshold is the presence of strong directional edges/patterns. These can be estimated from the $s_d$ parameters computed in Eq. (5) as

$$\delta = s_{d_{opt}} - s_{d_{ortho}} \ , \tag{14}$$

where $d_{ortho} = d_{opt} + 4 \pmod 8$. We compute the direction filtering threshold for each block as

$$T_d = T_0 \cdot \max\left(\tfrac{1}{2}, \min\left(3, \alpha_2 \delta^{1/6}\right)\right) \ , \tag{15}$$

where $\alpha_2$ also depends on the input scaling. For the second filter, we use a more conservative threshold that depends on the amount of change caused by the directional filter.

$$T_2\left(i,j\right) = \min\left(T_d, \tfrac{1}{3}T_d + \left|y\left(i,j\right) - x\left(i,j\right)\right|\right) \ . \tag{16}$$

As a special case, when the pixels corresponding to the $8 \times 8$ block being filtered are all skipped, then $T_d = T_2 = 0$, so no deringing is performed.

## V. Superblocks and Signaling

The filtering is applied one superblock at a time, in a way that depends on the level $\ell$. The level can take one of 6 values:

$$\ell \in \{0,\ 0.5,\ 0.7,\ 1.0,\ 1.4,\ 2.0\} \ , \tag{17}$$

where $\ell = 0$ disables the deringing filter for the current superblock. The level is the only information coded in the bitstream by the deringing filter. On keyframes, it is entropy-coded based on the neighbor values. On inter-predicted frames, the level is only coded for superblocks that are not skipped and is entropy-coded based on a single adapted probability distribution (no context from the neighbors). Superblocks where no level is coded have deringing disabled. Similarly, any skipped block within a superblock has deringing disabled, even if it is signaled enabled for the superblock.

The level of the deringing filter in AV1 is handled similarly, except that only four levels are currently available and there is no entropy coding yet.

The deringing process sometimes reads pixels that lie outside of the superblock being processed. When these pixels belong to another superblock, the filtering always uses the unfiltered pixel values – even for the second stage filter – so that no dependency is added between the superblocks. This makes it possible to filter all superblocks in parallel. When the pixels used for a filter lie outside of the viewable image, we set $f\left(d,T\right) = 0$ in Eq. (10).

## VI. Results

The deringing filter described here has been implemented for the Daala [8] codec and is available in the master branch of the Daala Git repository [9]. Its implementation lies in the `src/dering.c` file. An example of the effect of the deringing filter at low bitrate on a still image is shown in Fig. 6.

Table III
Deringing filter Bjøntegaard-delta [11] rate for still images (lower is better) in Daala.

| Bitrate (bpp) | PSNR | PSNR-HVS | SSIM | FAST-SSIM |
|---|---|---|---|---|
| 0.1 – 0.2 | -3.5% | -2.8% | -1.6% | +2.6% |
| 0.2 – 0.5 | -2.9% | -2.2% | -1.6% | +3.2% |
| 0.5 – 1 | -1.7% | -0.9% | -1.0% | +3.6% |

Table IV
Deringing filter Bjøntegaard-delta [11] rate for video sequences (lower is better) in Daala.

| Bitrate (bpp) | PSNR | PSNR-HVS | SSIM | FAST-SSIM |
|---|---|---|---|---|
| 0.005 – 0.02 | -5.8% | -4.3% | -4.0% | +7.2% |
| 0.02 – 0.06 | -7.4% | -4.7% | -5.8% | +11.5% |
| 0.06 – 0.2 | -7.9% | -5.0% | -7.7% | +12.2% |

We tested the deringing filter using the Are We Compressed Yet? [10] online testing tool. The results for still images are shown in Table. III for the subset1 test set and those for video are shown in Table IV for the ntt-short1 test set. Visual inspection confirms that the quality is greatly improved, despite the regression in the FAST-SSIM results.

## VII. Conclusion

We have demonstrated an effective algorithm for removing ringing artifacts from coded images and videos. The proposed filter is based on the conditional replacement filter (CRF) and takes into account the direction of the patterns it is filtering to reduce the risk of blurring. Objective results show a bit-rate reduction between 4% and 8% on video sequences.

## References


[1] H. S. Malvar and D. H. Staelin, "The LOT: Transform coding without blocking effects." *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, no. 4, pp. 553–559, 1989.

[2] T. Tran, J. Liang, and C. Tu, "Lapped transform via time-domain pre- and post-filtering," *Signal Processing, IEEE Transactions on*, vol. 51, no. 6, pp. 1557–1571, June 2003.

[3] J.-M. Valin and T. B. Terriberry, "Perceptual vector quantization for video coding," in *Proceedings of SPIE Visual Information Processing and Communication*, vol. 9410, 2015, pp. 941 009–941 009–11, `arXiv:1602.05209 [cs.MM]` http://arxiv.org/abs/1602.05209.


Table V
Deringing filter Bjøntegaard-delta [11] rate for still images (lower is better) in AOM codec.

| Bitrate (bpp) | PSNR | PSNR-HVS | SSIM | FAST-SSIM |
|---|---|---|---|---|
| 0.2 – 0.5 | -2.0% | -1.4% | -0.9% | +2.8% |
| 0.5 – 1 | -0.9% | -0.5% | -0.6% | +1.5% |

Table VI
Deringing filter Bjøntegaard-delta [11] rate for video sequences (lower is better) in AOM codec.

| Bitrate (bpp) | PSNR | PSNR-HVS | SSIM | FAST-SSIM |
|---|---|---|---|---|
| 0.02 – 0.06 | -2.5% | -1.5% | -1.5% | +3.8% |
| 0.06 – 0.2 | -2.0% | -0.8% | -1.3% | +3.1% |

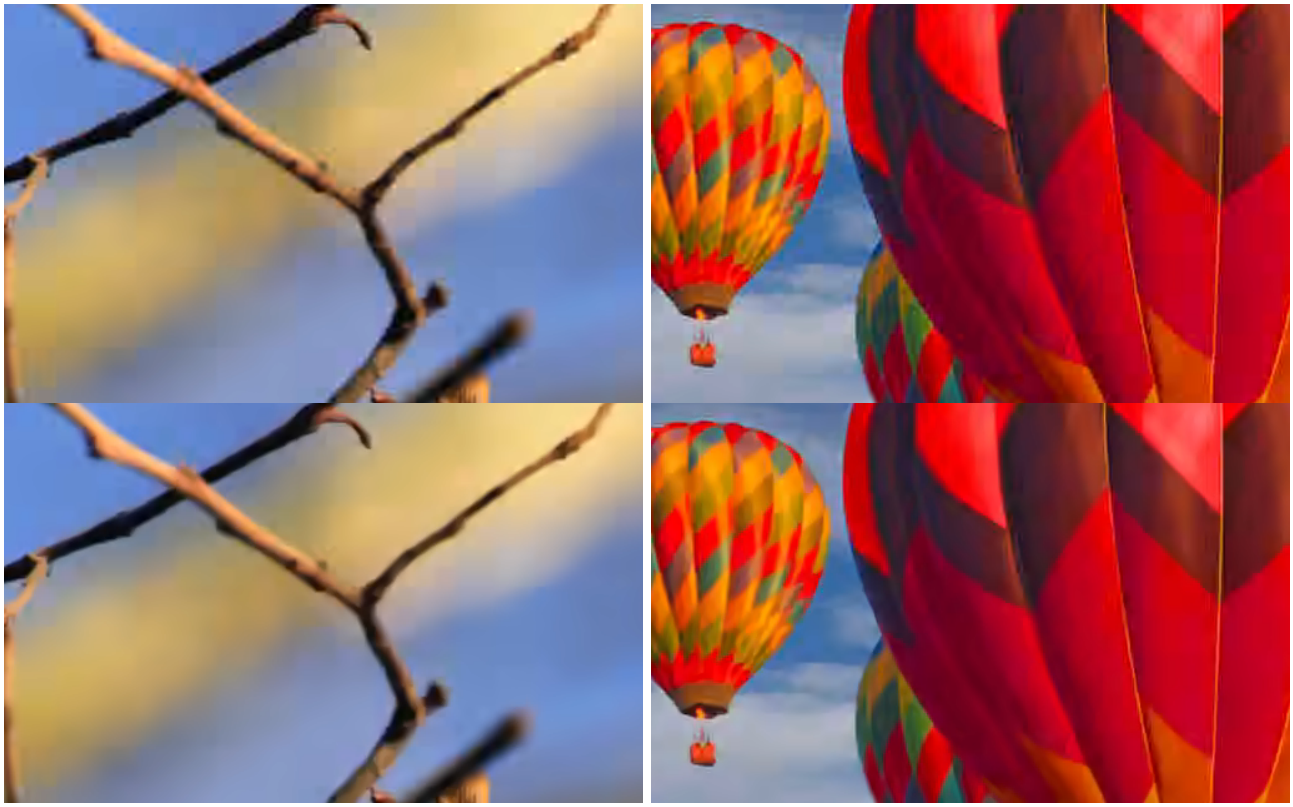

Figure 6. Visual effect of deringing at low bitrate for Daala. Top: without deringing, bottom: with deringing.


[4] C. M. Fu, E. Alshina, A. Alshin, Y. W. Huang, C. Y. Chen, C. Y. Tsai, C. W. Hsu, S. M. Lei, J. H. Park, and W. J. Han, "Sample adaptive offset in the hevc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1755–1764, Dec 2012.

[5] "Median filter," https://en.wikipedia.org/wiki/Median_filter.

[6] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of IEEE International Conference on Computer Vision*, 1998.

[7] J.-M. Valin, "A deringing filter for daala... and beyond," https://people.xiph.org/~jm/daala/deringing_demo/, 2016.

[8] "Daala website," http://xiph.org/daala/.

[9] "Daala git repository," http://git.xiph.org/?p=daala.git;a=summary.

[10] "Are we compressed yet?" https://arewecompressedyet.com/.

[11] J. M. T. Daede, "Video codec testing and quality measurement," https://tools.ietf.org/html/draft-daede-netvc-testing, 2015.