REAL-TIME VIDEO DENOISING ON MOBILE PHONES

Jana Ehmann, Lun-Cheng Chu, Sung-Fang Tsai and Chia-Kai Liang

Google Inc.

ABSTRACT

We present an algorithm for real-time video denoising on mobile platforms. Based on Gaussian-Laplacian pyramid decomposition, our solution's main contributions are fast alignment and a new interpolation function that fuses noisy frames into a denoised result. The interpolation function is adaptive to local and global properties of the input frame, robust to motion alignment errors, and can be computed efficiently. We show that the proposed algorithm has comparable quality to offline high-quality video denoising methods, but is orders of magnitude faster. On a modern mobile platform, our work takes less than 20ms to process one HD frame, and it achieves the highest score on a public benchmark.

Index Terms— video denoising, real-time processing, mobile platforms, pyramid decomposition

1. INTRODUCTION

Videography on mobile phones has become a part of our everyday lives due to its convenience. Phone cameras are used for recording, video chatting, and even live streaming. However, the quality of mobile videos is still inferior to the ones taken by professional cameras, especially in low light conditions. Mobile imaging sensors are much smaller and noisier, and it is difficult to remove temporal noise using single frame spatial denoising methods. Although videos include temporal information for effective denoising, advanced video denoising algorithms are too slow for mobile phones: most of them are not even real-time on desktops (Sec. 2).

In this paper, we present a novel highly effective and efficient temporal denoising method. First, we propose a new non-linear filter to address spatially varying noise and motion compensation error. Second, we show that an optimized one-tap temporal recursive filtering can achieve competitive denoising quality. Third, we design the motion estimation and filtering blocks such that they share the bulk of the computation by using a common image pyramid representation.

Our algorithm can achieve 30+ fps performance on HD resolution on modern mobile platforms, with denoising quality that is comparable to the state-of-the-art offline methods. We integrate the proposed algorithm into the image processing pipeline on Google Pixel 2, which obtained the highest rating in video quality on the public DxO benchmark [1].

2. BACKGROUND

Video denoising is a mature topic that has been extensively studied and we only briefly review the relevant work here.

The early methods extend non-local means [2], bilateral filtering [3] or wavelet denoising [4], but do not consider frame or object motions explicitly. VBM3D [5] and VBM4D [6] extend BM3D [7] with temporal block alignment and grouping, and [8] extend 3D DWT with motion compensation. Some methods use dense optical flow [9, 10] to further improve their results.

Note that none of these methods are designed for real-time mobile applications. Even on the desktop, most other techniques cannot achieve real-time performance at CIF resolution (let alone HD) unless a powerful GPU is used. More efficient methods use a single aligned block per frame [11, 12, 13]. Nonetheless, they need more frames to gather enough blocks, and are still too slow for mobile video recording.

3. REAL-TIME VIDEO DENOISING

Our algorithm has two main stages: an *alignment* stage that computes the displacement map between two consecutive frames and a *merging* stage to combine the input frame and previous frame's output. The result of the merging is used as one of the inputs to the next frame's merging stage. Note that our algorithm does not perform spatial filtering to save computational time, although the pyramid framework implicitly makes it a spatio-temporal method. With proper alignment, even pure recursive temporal filtering can converge to a very low-noise result [14].

The flowchart is shown in Figure 1. Frame at time t is decomposed into a Gaussian and a Laplacian pyramid. The Gaussian pyramids are used in alignment (Sec. 3.1), while the Laplacian ones are used in merging (Sec. 3.2). The merged pyramid is collapsed to create the final output. We implement the algorithm with CPU/GPU and model the real noise on the mobile platform (Sec. 3.3).

In following sections, $\mathcal{G}^g$ denotes a Gaussian pyramid level $g \in [0, ..., N_G]$, $\mathcal{L}^l$ denotes a Laplacian pyramid level $l \in [0, ..., N_L]$, with N_G and N_L denoting pyramid heights, and pixel location within a given pyramid level is given by $\mathbf{p} = (x, y)$. The noise variance map for each Laplacian pyramid level is given in $n^l(\mathbf{p})$.

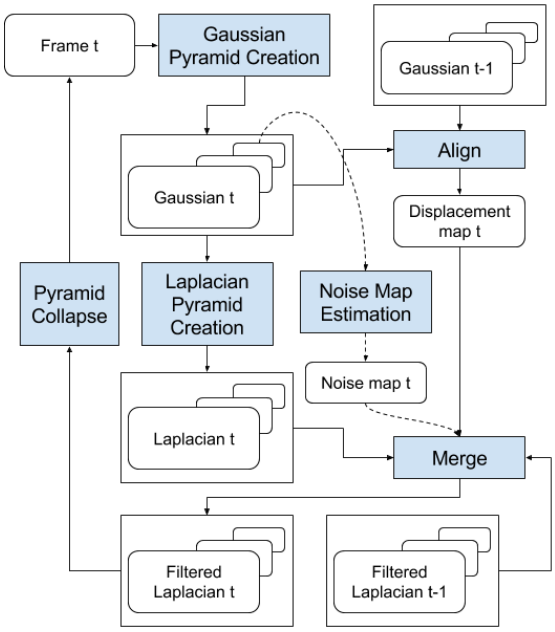

Fig. 1: Flowchart of proposed algorithm. Blue blocks are processing units and white ones are image buffers.

3.1. Alignment

We use a hierarchical method to compute a displacement map $A(\mathbf{p})$ with a large search range. Our algorithm is based on the fast optical flow method of [15] with several optimizations. We also perform iterative inverse Lucas-Kanade search at each $\mathcal{G}^g$, but restrict the motion vector computation to vertices of a coarse grid with a step size of 16-pixels in each dimension.

In our experiments, we found that integer-precision motion vectors at the coarser pyramid layer are accurate enough to initialize the motion vector search for the next finer layer. Hence, we can remove all unnecessary interpolation for sub-pixel alignment error calculation, except for the final layer. Finally, to increase the motion search range and reduce the iteration count, we use the horizontal and vertical 1D projections at the coarsest layer to estimate the global motion by cross correlation. Our optimized method is 4x faster than the reference one in [15].

3.2. Merging

In the merging stage, we combine the current Laplacian pyramid with the Laplacian pyramid of the previous frame. The cornerstone of our proposed algorithm is the fact that we perform IIR processing, meaning that we use the previously filtered result as opposed to the non-processed previous frame. The IIR formulation has stronger denoising properties, but has the drawback of potential artifact propagation. We designed our interpolating function to be robust to alignment errors such that, even in case of erroneous alignment artifacts like ghosting, non-existing features or excessive blurring are not introduced.

We first use the displacement map to form the aligned pyramid $\mathcal{L}_a^l$ using the previous pyramid $\mathcal{L}_p^l$:

$$\mathcal{L}_a^l(\mathbf{p}) = \mathcal{L}_p^l(\mathbf{p} + A^l(\mathbf{p})), \tag{1}$$

where A^l is a scaled version of A at level l.

The resulting pyramid layer $\mathcal{L}_r^l$ is calculated as an interpolation between the current layer $\mathcal{L}_c^l$ and $\mathcal{L}_a^l$:

$$\mathcal{L}_r^l(\mathbf{p}) = \mathcal{I}_c(\mathbf{p}) \cdot \mathcal{L}_c^l(\mathbf{p}) + \mathcal{I}_p(\mathbf{p}) \cdot \mathcal{L}_a^l(\mathbf{p}), \tag{2}$$

where $\mathcal{I}_c$ and $\mathcal{I}_p$ are interpolating functions which adapt to pixel locations, values, and noise levels.

To better explain how the intepolating functions are designed, let us denote $\mathcal{L}_\Delta^l(\mathbf{p}) = \mathcal{L}_c^l(\mathbf{p}) - \mathcal{L}_a^l(\mathbf{p})$ as the pixel value difference. By rearranging Eq. 2 we obtain:

$$\mathcal{L}_r^l(\mathbf{p}) = w_c^l \cdot \mathcal{L}_c^l(\mathbf{p}) + w_p^l \cdot (\mathcal{L}_a^l(\mathbf{p}) + \mathcal{I} \cdot \mathcal{L}_\Delta^l(\mathbf{p})). \tag{3}$$

In this formulation, w_c^l is the minimum value of $\mathcal{I}_c$, w_p^l is the maximum value of $\mathcal{I}_p$, and $\mathcal{I}$ is the interpolation factor which determines the final weights for current and previous pixel. We explain these values in detail below.

3.2.1. Interpolation bounds w_c^l and w_p^l

The denoising strength limits in our algorithm are defined by w_c^l and w_p^l. If we set $w_c^l \leq w_p^l$, we can achieve stronger denoising power by emphasizing the previously filtered component. Note that because they are functions of pyramid level, we can choose to perform stronger denoising in certain frequency bands, where we are more confident that we would have less artifacts from alignment errors.

Due to averaging, high-frequency features such as textures and edges sometimes get smoothed out if the alignment is off even by few pixels. To compensate for this issue, we make one key modification to interpolation weights:

$$w_c^l + w_p^l \geq 1. \tag{4}$$

By effectively boosting the weighted average with a factor slightly above 1, we can restore high-frequency features while retaining the denoising strength.

In practice, we adaptively adjust w_c^l and w_p^l. For well-lit scenes, we can keep enough details without enhancing the higher frequencies, while for dimly lit scenes, the oversharpening is less visible, and we can safely raise the amplitude of higher frequencies.

3.2.2. Interpolation factor $\mathcal{I}$

The design of $\mathcal{I}$ is the key differentiator of our algorithm. It takes into account various spatio-temporal effects in order to achieve good denoising strength with minimal artifacts. This is done by considering the amount of expected noise and the actual difference between two pyramid values. We also consider the patch-wise alignment error – for cases where we do not have a good matching patch in the previous frame and we need to be more conservative in order to not introduce any artifacts to the current frame.

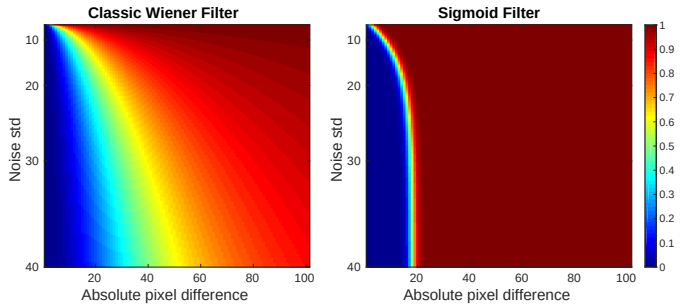

Fig. 2: Interpolator functions.

The value of $\mathcal{I}$ always lies in the range of [0, 1]. As we can see in Eq. 3, higher $\mathcal{I}$ would lower the denoising strength as the contribution from previous frame is reduced. We compute two interpolation factor candidates and pick the larger one: $\mathcal{I} = \max(\mathcal{I}_e, \mathcal{I}_\Delta)$, where $\mathcal{I}_e$ is based on the alignment error and $\mathcal{I}_\Delta$ is based on pixel difference.

In regions with good alignment accuracy, $\mathcal{I}_\Delta$ dominates over $\mathcal{I}_e$. For such pixels, we can confidently interpolate between the current and previous pyramid. However, in areas where the alignment error is large, $\mathcal{I}_e$ would dominate:

$$\mathcal{I}_e = \max(1, A_e^l(\mathbf{p}) \cdot C_e), \tag{5}$$

where $A_e^l(\mathbf{p})$ is the block matching error from the alignment stage. The tuning parameter C_e sets the limit of acceptable motion alignment error. $\mathcal{I}_e$ also turns off denoising in areas with gross alignment error due to occlusions, objects moving in and out of frame, etc.

For $\mathcal{I}_\Delta$, one popular choice is the well-known Wiener filter:

$$\mathcal{I}_{Wiener} = \frac{\mathcal{L}_\Delta^l(\mathbf{p})^2}{\mathcal{L}_\Delta^l(\mathbf{p})^2 + n^l(\mathbf{p})}. \tag{6}$$

It has been used in multi-frame Fourier and wavelet denoising [13, 16] and worked reasonably well when the alignment accuracy is high. However, due to higher dynamics in the scene and limited computational budget in video recording, the alignment confidence is usually lower, and we have to be more conservative because of anticipation of alignment errors. Therefore, we propose a new interpolator that also depends on the noise variance and the difference between pixel values, but has a sigmoid shape instead:

$$\mathcal{I}_\Delta(\mathbf{p}) = (1 + \exp^{-(|\mathcal{L}_\Delta^l(\mathbf{p})| - m)})^{-1}, \tag{7}$$

$$m(\mathbf{p}) = 1 + C_{middle} \cdot (1 - \exp^{-n^l(\mathbf{p}) \cdot C_{noise}^l}), \tag{8}$$

where m is the middle point (i.e., interpolating factor is 0.5 when $|\mathcal{L}_\Delta^l(\mathbf{p})| = m(\mathbf{p})$), C_{middle} defines the boundary for the bending (middle) point, and C_{noise}^l are appropriate noise scaling constants for different pyramid levels.

The values of $\mathcal{I}_{Wiener}$ and $\mathcal{I}_\Delta$ for different noise levels and absolute pixel differences can be seen in Fig. 2. Even with low noise levels and high pixel differences, $\mathcal{I}_{Wiener}$ still allows the previous pixel to contribute significantly to the result, and leads to ghosting or over-smoothing artifacts.

On the other hand, $\mathcal{I}_\Delta$ has a sharp phase transition. It applies stronger denoising in higher noise levels by allowing the interpolation factor to be near 0 for a wider range of values than

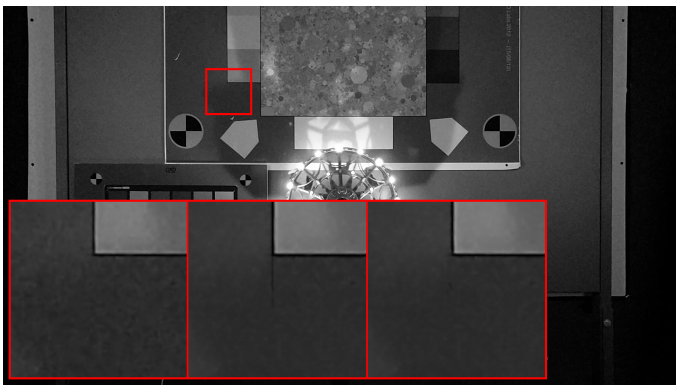

Fig. 3: Comparison of different interpolators. Background: original noisy frame. Zoom-in: original block (left), denoised with Wiener filter (center), denoised with proposed sigmoid interpolator (right).

$\mathcal{I}_{Wiener}$. However, even for very high noise levels, large pixel differences mostly occur due to wrong alignment as opposed to noise, and $\mathcal{I}_\Delta$ quickly blocks the contributions from the previous frame to avoid artifacts. Fig. 3 clearly shows how the proposed $\mathcal{I}_\Delta$ better suppresses the artifacts compared to $\mathcal{I}_{Wiener}$, while maintaining the same denoising level in the rest of the frame.

After all the Laplacian pyramid layers $\{\mathcal{L}_r^l\}$ have been denoised, we simply collapse them into the output frame, and store them as $\{\mathcal{L}_p^l\}$ of the next frame.

3.3. Implementation Details

We implement our algorithm with CPU/GPU co-processing and test it on both the desktop and the Google Pixel 2, which uses Qualcomm Snapdragon 835. We implement OpenGLES shaders for the pyramid creation and merge on GPU, and implement alignment and noise estimation on CPU using Halide [17]. While the CPU-only implementation can still achieve real-time throughput, our joint CPU/GPU solution greatly reduces the power consumption (-66%). We find that the existing spatial denoiser is effective enough for chroma channels, and only enable our algorithm on the luma channel.

3.3.1. Noise Estimation

Real noise characteristics vary with both sensor and scene properties [18], and proper noise modeling is critical for denoising performance [13]. We model the noise variance by supplying a noise map $n^l(\mathbf{p})$ at every layer for each pixel.

In our target platform, we use controlled lighting and camera settings to calibrate the sensor noise and determine how the image processing stages (lens shading correction, white balance, color space conversion, tone mapping, gamma correction etc.) transform the image and the noise variance. At capture time, we use the available auxiliary information about camera settings, and inverse transform the coarsest layer of $\mathcal{G}^g$ back to the raw sensor domain, and transform the sensor noise to obtain $n^l(\mathbf{p})$.

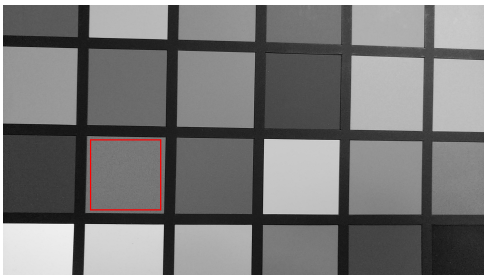

Fig. 4: Static test scene, with the ROI marked in red.

Method	PSNR	Temporal var.	Runtime (s)
Source	37.02	13.10	-
3DWF [11, 12]	40.56	5.66	0.25 (desktop)
VBM3D [5]	45.92	1.40	5.40 (desktop)
HDR+ [13]	39.79	6.83	0.15 (Pixel 2)
Proposed	43.71	2.55	0.018 (Pixel 2)

Table 1: Mean PSNR (dB) and temporal variance within the ROI, and processing time per-frame of denoising methods.

Mobile Phone	Texture	Noise	Overall
Google Pixel 2	57	80	96
Huawei Mate 10 Pro	48	77	91
Apple iPhone X	51	66	89
Galaxy Note 8	47	73	84

Table 2: DxOMark scores related to video denoising among 2017 top performers (higher is better).

4. RESULTS

Comparing our method to state-of-the-art is yet another challenge. We have developed an algorithm which was deployed in a mobile commercial product, and therefore its performance compared to the academic solutions may fall short due to the very strict computational, power, and memory budgets. On the other hand, other commercial solutions are available to us only in their final systems form, i.e., we cannot perform the standard testing by supplying the exact same input to various algorithms and evaluating the results. Therefore we present various set-ups: lab tests, real-world evaluation, as well as a commercial benchmarking protocol.

For the lab experiment, we recorded a short video (74 frames, 1080p resolution) on a tripod containing a static scene of a colored checkerboard and used the temporal average of all frames as the ground truth. In addition to our own, we applied three representative temporal denoising algorithms to this video: VBM3D [5], 3DWF [11, 12], and HDR+ [13]. In Table 1, we list the runtime along with the mean PSNR and the temporal variance of a flat region highlighted in Fig. 4. Among evaluated algorithms, VBM3D achieves best quality, but it takes several seconds to process one frame using a powerful desktop CPU (Intel Xeon E5 in HP z840 workstation). The proposed method is orders of magnitude faster and achieves sustainable real-time performance on Google Pixel 2 with one thread (Qualcomm Snapdragon 835). Also, our method achieves competitive results both for PSNR and temporal variance.

Fig. 5 shows real world samples of low-light videos recorded by Pixel 2. Note that these are much more challenging than normal lab tests: the phone is handheld without tripod, the scenes can include moving objects, and the lighting conditions may vary. The proposed method can greatly reduce noise in all these conditions without affecting details.

(a) Night

(b) Person

(c) Stair

Fig. 5: Field videos taken by Google Pixel 2. In each frame the video denoising is disabled at the right half part, and the middle/right columns are the zoom-in around vertical center. (Best reviewed in the electronic version.)

Finally, we list the public DxO benchmark results in Table 2. The DxO Lab performs both lab and field tests in different scenes (charts, natural objects, selfies, group shots...) and under different lighting conditions (walking, panning, outdoors, indoors, very low light conditions, high dynamic range...) and combines several subjective and objective metrics into a report [19, 1]. It is the most popular benchmark for phone camera quality assessment. We can see that Pixel 2 achieves the highest video noise score and preserves more texture than other phones, which all have their own proprietary spatial and temporal denoising solutions. The overall video score is the highest among all premium 2017 smartphones.

5. REFERENCES

[1] "DxOMark mobile test protocol and scores," www.dxomark.com/dxomark-mobile-testing-protocol-scores/, Accessed: 2018-01-18.

[2] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Processing Letters*, vol. 12, no. 12, pp. 839–842, 2005.

[3] E. P. Bennett and L. McMillan, "Video enhancement using per-pixel virtual exposures," in *ACM TOG*, 2005, vol. 24, pp. 845–852.

[4] H. Malm, M. Oskarsson, E. Warrant, P. Clarberg, J. Hasselgren, and C. Lejdfors, "Adaptive enhancement and noise reduction in very low light-level video," in *ICCV*, 2007.

[5] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3d transform-domain collaborative filtering," in *2007 15th European Signal Processing Conference*, 2007, pp. 145–149.

[6] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, "Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms," *IEEE TIP*, vol. 21, no. 9, pp. 3952–3966, 2012.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE TIP*, vol. 16, no. 8, pp. 2080–2095, 2007.

[8] S. Yu, M. O. Ahmad, and M. Swamy, "Video denoising using motion compensated 3-D wavelet transform with integrated recursive temporal filtering," *IEEE TCSVT*, vol. 20, no. 6, pp. 780–791, 2010.

[9] C. Liu and W. T. Freeman, "A high-quality video denoising algorithm based on reliable motion estimation," in *ECCV*, 2010, pp. 706–719.

[10] A. Buades, J. L. Lisani, and M. Miladinovic, "Patch-based video denoising with optical flow estimation," *IEEE TIP*, vol. 25, no. 6, pp. 2573–2586, 2016.

[11] A. Kokaram, "3D Wiener filtering for noise suppression in motion picture sequences using overlapped processing," in *Signal Processing V, Theories and Applications*, 1994, pp. 1780–1783.

[12] A. Kokaram, D. Kelly, H. Denman, and A. Crawford, "Measuring noise correlation for improved video denoising," in *ICIP*, 2012, pp. 1201–1204.

[13] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy, "Burst photography for high dynamic range and low-light imaging on mobile cameras," *ACM TOG*, vol. 35, pp. 192:1–192:12, 2016.

[14] T. Hachisuka, S. Ogaki, and H. W. Jensen, "Progressive photon mapping," *ACM TOG*, vol. 27, no. 5, pp. 130:1–130:8, 2008.

[15] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast optical flow using dense inverse search," in *ECCV*, 2016, pp. 471–488.

[16] V. Zlokolica, A. Pizurica, and W. Philips, "Wavelet-domain video denoising based on reliability measures," *IEEE TCSVT*, vol. 16, no. 8, pp. 993–1007, 2006.

[17] J. Ragan-Kelley, A. Adams, S. Paris, M. Levoy, S. Amarasinghe, and F. Durand, "Decoupling algorithms from schedules for easy optimization of image processing pipelines," *ACM TOG*, vol. 31, no. 4, pp. 32:1–32:12, 2012.

[18] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image," *IEEE TPAMI*, vol. 30, no. 2, pp. 299–314, 2008.

[19] F. Cao, F. Guichard, and H. Hornung, "Dead leaves model for measuring texture quality on a digital camera," in *Digital Photography*, 2010, p. 75370.