CODING TRAJECTORY: ENABLE VIDEO CODING FOR VIDEO DENOISING

Zhihang Ren, Peng Dai, Shuaicheng Liu, Shuyuan Zhu, Bing Zeng

Institute of Image Processing
University of Electronic Science and Technology of China

ABSTRACT

We introduce a novel video denoising approach which can produce a clean video by utilizing redundant image patches existed in the video frames. Previous multi-frame video denosing approaches either require image registration or employ Patch Match algorithms for the discovery of the patch redundancy. However, these computations are time-consuming and prone to errors. On the other hand, nearly all captured videos have been compressed. Such a compression can produce a rich set of block-based motion vectors that can be utilized for the redundant patch extraction, leading to the efficient video denosing. To be specific, the motion vectors and frame references can be obtained from the video coding. Given a noised frame block, we follow its motion vectors from the coding to form a trajectory and gather a set of block candidates along the routes from its nearby frames. The trajectory is referred to as *Coding Trajectory*. Then, the corresponding denoised block is generated by weighted fusing the block candidates with outlier rejections. A denoised frame is consisted of all the denoised blocks. We compare our method with several state-of-the-art approaches, such as VBM3D and VBM4D, in terms of PSNR and SSIM. The experiments show that our method can achieve high quality results while runs much faster then the other approaches.

***Index Terms*—** Video Coding, video denoising, coding trajectory

1. INTRODUCTION

Video denosing is a classic problem that has attracted lots of attentions from the research community during the past decades [1, 2, 3, 4] . Videos captured in a dimly-lit environment often suffer from strong noises that severely lowers the quality of the videos. Video denoising approaches target at reducing the noises such that the video quality can be largely improved. With respect to the multi-frame video denosing, Liu *et al.* proposed to denoise one image by several images that being captured by a cell-phone under the burst mode [5]. Ren *et al.* proposed to denoise videos by exploring the frame registration using mesh-based motion models [6]. [1]

[1] http://www.liushuaicheng.org/ICIP/2018/CodingTrajectory.mp4

In general, the challenge of a practical video denosing method lies in two aspects. First, the quick discovery of similar image patches resided in nearby adjacent video frames. Second, the robust patch fusing for denoising with outlier rejection. Previous approaches [5, 6] often extract patches with similar contents from aligned sequence. However, the alignment requires detecting image features [7] and manipulate motion models, such as the homography-flow motion model proposed in [5] and the meshflow motion model [8] adopted in [6]. The image alignment is time consuming and prone to errors in many consumer level videos. The alignment is important for some applications such as the video deblurring [9], the HDR [10] and the video stabilization [11]. With respect to the video denosing, however, it is unnecessary to extract similar patches by aligning images. The similar patches can be found by search around the frame and adjacent frames using the PatchMatch [12, 13, 14].

In this paper, instead of conducting PatchMatch which still requires some computations, we directly use the block motion vectors extracted from the video coding. The video coding aims at removing spatial and temporal redundancies of the raw video data. Nearly all captured videos have been compressed before the storage and the distribution. However, after the decoding, the block motion vectors are usually discarded. We notice that these motion vectors can effectively link similar patches across different frames for the video denosing. Specifically, given a patch, we collect a set of patches by following the routes of motion trajectories embedded in the video coding. Both B-frame and P-frame in the video coding can come up with trajectories, which yields sufficient number of patches for the denoising. We fuse these patches with weights derived from patch similarities. We further reject inconsistent pixels caused by dynamic contents. In this way, the video can be denoised with a very low computational cost. Moreover, it is convenient to implement the method into various platforms (such as the embedded systems), because the coding unit is almost compulsory as long as the platform supports video applications. Various challenging examples demonstrate the effectiveness of the proposed method.

The rest of the paper is arranged as follows. Section 2 reviews the related work. Section 3.1 and Section 3.2 illustrate our algorithm in detail. In Section 4, the experimental results are demonstrated and discussed. In particular, the results

of classical algorithms are put into comparison to illustrate the efficiency and quality of our algorithm. Finally, Section 5 concludes this paper.

2. RELATED WORKS

2.1. Video Coding

Video coding aims at compressing the data size by removing the redundancies (spatial and temporal) in the raw video data. The intra coding and the inter coding are designed to remove such redundancies. Specifically, this removing operation is block based. In both coding modes, each image block is usually compressed by the discrete cosine transform (DCT) [15], quantization, and entropy coding.

In the redundancy-removing procedure, DCT is even more important as it transforms the pixel domain into the frequency domain, in which the block energy is much more compacted into only a few coefficients. In order to further improve the performance, recently, several new transforms, such as the directional DCT (DDCT) [16] and some novel unitary transforms [17], have been proposed. Meanwhile, the integer DCT [18] has been developed to support the hardware-based implementation.

At the same time, it has been convinced that the motion-compensated prediction (MCP) [19] could largely promote the efficiency of the temporal redundancies removing. Thus, MCP has been adopted in all video codecs to construct the inter coding mode. The motion estimation (ME) searches for the best MV. The most obvious method to find the optimal MV is to do a tedious search. Therefore, to solve this problem, lot of algorithms have been developed, such as the three-step search (TSS) [20], the fourstep search (4SS) [21], and the diamond-shaped search [22].

2.2. Video Denoising

Image/video denoising approaches [1, 2, 3, 4] focus on removing or suppressing noises for the video quality improvement. Some methods adopted non-local measurements for the image restoration [14] while others explore the sparsity of a multiresolution representation in the wavelet domain [23]. In general, multiple image denoising can provide superior results compared with single image approaches as more observations are provided. Video frames often contain complementary information that can be merged for denoising [24].

With respect to the single image approach, the BM3D algorithm utilized a sliding-window manner and block-matching to form 3D arrays. Then it denoised the image by filtering it in a 3D transform domain [4]. The VBM3D method [25] extended the BM3D approach to denoise the videos. Based on the highly sparse signal representation in a local 3D transform domain, the VBM3D algorithm could denoise videos efficiently by aggregating the matched blocks as weighted averages. The BM4D method utilized cubes of voxels, which could be stacked into a four-dimensional group, and jointly be filtered in the transformed domain [26]. The VBM4D [27] achieved the high quality performance by tracking the motion vectors of matched blocks and staking them along the trajectory. The results could be obtained by the collaborative filtering. Different from the VBM4D, we directly obtain motion vectors embedded in the video coding for an efficient solution.

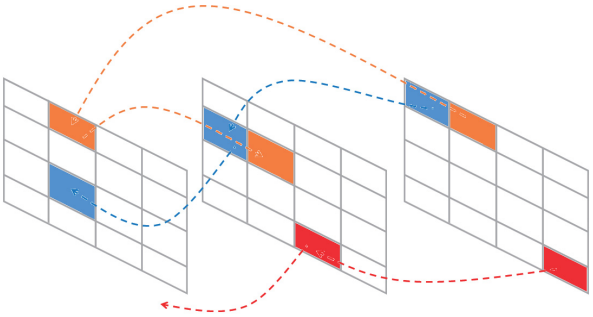


Fig. 1. Coding Trajectory. Different colors indicate different trajectories. (large grid size for clearer illustration).

3. OUR METHOD

We describe our approach based on the H.264/AVC framework for the convenience of its relatively simpler structure as compared with the H.265/HEVC framework. Note that the same processing is also hold for the H.265/HEVC.

3.1. Block Tracking

There are three types of frames in the video coding, including I-frame, P-frame and B-frame. The I-frame is regarded as the starting frame, which is referred by the P-frame and the B-frame. The P-frame can only refer to its previous frames while the B-frame can go either directions. In our case, the first video frame is set to I-frame and we alow both P-frame and B-frame references.

In H.264/AVC, each frame is processed in units of macroblocks with size 16×16. Each macroblock can be encoded with *intra* or *inter* mode. A tree-structured motion compensation strategy is supported, where each macroblock can be further divided into smaller sizes, including 16×8, 8×16, 8×8, 8×4, 4×8 and 4×4. The divison is performed by the encoder using a rate-distortion optimization (RDO). For image regions with less textures, a large division may be desired to decrease the coding bits while for regions with rich textures, a smaller decision is required to capture the image details. In our implementation, we always fix the units division as the smallest 4×4. If a block have a larger size, we duplicate its motion vector and assign them to its containing

4×4 blocks. The fixed 4×4 division is convenient to generate the coding trajectory.

Figure 1 shows an example of the block tracking as well as the coding trajectory. After receiving the compressed bitstream, we can easily extract all motion vectors, with information such as its referenced frame and its motion direction and magnitude. As shown in Fig. 1, each block can refer to its previous frame (P-frame and B-frame) and its future frame (B-frame). Starting from a block, we trace its motion to the next block and push blocks along the trail into our block candidate list. For blocks which contains two motions, we randomly chose one of them to form our trajectory. Note that, the motion vector is an integer value and cannot be exactly landed within the block center. When we conduct the 'jump', the next motion vector should take the bilinear interpolation of the motion vectors from its surrounding four blocks. In our implementation, we jump 6 times to collect 6 patch candidates.

Sometimes, the interpolated motion vector may jump outside the frame border (the red arrow shown in Fig. 1). In such cases, the tracking should be terminated. Moreover, there are also chances that the tracking may touch the border of the video (the first frame or the last frame). For these situations, the tracking should also be terminated.

3.2. Fusing

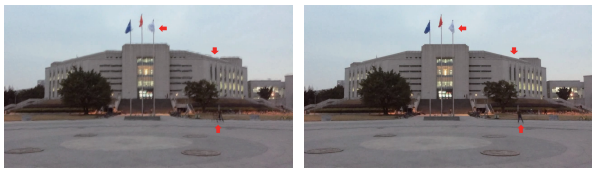
3.2.1. Block weights

Block tracking in Section 3.1 illustrates how to form coding trajectories to collect the block candidates. In this section, the fusing strategy will be described, including the weight calculation based on the block similarity in terms of the SSIM and the pixel consistency verification.

Given the target noise block, the tracked candidates may not be the blocks with the very similar contents due to the interferences of the noises. Fusing these inaccurate blocks could result in many uncomfortable effects, such as the ghosting and the blurring of the structures. Therefore, during the block fusing, the weights of the candidate blocks should be carefully designed so that the artifacts could be attenuated. In our algorithm, we choose the structural similarity index (SSIM) to indicate the similarity of the candidate block with the target block.

More specifically, the range of SSIM is within -1 and 1, in which the value 1 is only reachable in the case of two identical blocks. The positive value means these two blocks are positively correlated and vice versa. The magnitude of the SSIM indicates the correlated level in a positive manner, where 1 indicates identical and 0 indicates independent. More detailed properties and the proof of SSIM bound could be found in [28]. In our algorithm, due to the presence of the strong noise, the SSIM is always below 1. And according to the experiments, most SSIMs fall into the positive range. We compare the candidate block and the target block in terms of the SSIM and assign the weight w_b to the candidate according to the following strategy:

$$\omega_b = \begin{cases} 100 * SSIM & \text{if } SSIM \geqslant 0 \\ 0.01 & \text{if } SSIM < 0 \end{cases} \quad (1)$$



Without Consistency With Consistency

Fig. 2. Without (left) and with (right) consistency check. Please notice the strong ghosting effects in the left example of without the consistency.

3.2.2. Consistency check

Although the coding trajectory is conducted for every 4×4 blocks, we extract a relatively larger block size, 12×12, from the coding trajectory for the fusing. The bigger block size can lead to larger overlaps, which give raise to higher capabilities of the denoising after the fusing. We further add a pixel-level consistent check upon the block-level weights for the improved accuracy. The pixel-level consistent check can better handle the dynamic objects, such as pedestrians, bicycles and cars within the block, without which the ghosting effects can be easily introduced. On the other hand, we might avoid the artifacts by assigning the candidates with a lower block weights. However, the lower block weights could decrease the effects of the denoising.

Specifically, we establish a threshold check for each pixel in every candidate block. The threshold is set to 20 for each channel of the pixel (pixel ranges from $0 \sim 255$). So, after the consistency check, we can obtain a 12×12 mask K_i. Finally, the denoised block B_{final} is computed as:

$$B_{final} = \frac{\sum_{i=0}^{N} \omega_b K_i B_i}{\sum_{i=0}^{N} \omega_b K_i}, \quad (2)$$

where B_0 is the starting block, N is the tracking length and B_i is the block after i^{th} tracking. In our algorithm, the tracking length N is set to 6. Applying this operation for each block in all video frames, we could eventually obtain the final denoised result. Fig. 2 shows an example of with and without the consistency check. Please notice the ghosting effects of the moving person and flags, as well as the static building highlighted by the red arrow.

4. RESULTS

We run our method on an Intel i7 3.4GHz CPU with 32G RAM. Our unoptimized and unparalleled C++ implementa-

	$\sigma_{noise} = 5$			$\sigma_{noise} = 10$		
	flower	salesman	tennis	flower	salesman	tennis
Ours	0.106	0.136	0.096	0.099	0.128	0.090
BM3D [4]	0.337	0.548	0.482	0.353	0.566	0.516
VBM3D [25]	1.32	2.30	1.80	1.11	2.26	1.71
BM4D [26]	87	105.6	85.5	87.5	104.9	87
VBM4D [27]	115.5	137.9	125.0	116.6	139.6	128.9

Table 1. Comparison of processing time (seconds) for various algorithms under different levels of noise.

tion can achieve 1620ms on average to process a frame with resolution of 1280×720. The method can be further accelerated by the GPU, especially for the extraction of motion vectors and the fusing of candidate blocks which are highly parallelizable. Specifically, we spend 437 ms for calculating the SSIM weights, 485 ms for tracking the trajectory and consistency check, 698 ms for candidates averaging, respectively. Fig 4 gives a typical denoised result of our algorithm on real life data. Please refer to the supplementary file for the visual comparisons (the link at the first page).

Fig. 3. Examples in the evaluation.

We evaluate the performances on the synthetic data. (The examples are synthesised by clear ground truth videos and the Gaussian noise) Fig. 3 shows these examples. The resolutions are 352×240 for the 'flower', 352×288 for the 'salesman', 352×240 for the 'tennis', and 352×320 for the 'carpark'.

Fig. 4. An example of input (left) and denoised result (right).

4.1. Speed comparison

The results of running time for different methods on various samples are summarized in Table 1. We synthesize noises in two different levels in this experiment. As illustrated, our algorithm shows the highest efficiency, with the speed tens and hundreds of times faster than the speeds of the others. Note that the resolution of the test data is quite small compared to the real data which is much larger. Thus, our algorithm will be far more efficient than those algorithms when processing the real data.

4.2. Quality Comparison

We evaluate the performance on the example 'carpark'. We calculate the PSNR and the SSIM for the objective comparison. The results are reported in Fig. 5. As suggests, our algorithm can achieve comparable performance while running much faster than the other approaches. Also, notably, as shown in Fig 4, the denoising result is apparent(see the white wall of the building and the yellow light).

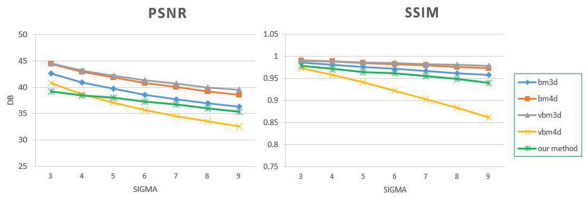


Fig. 5. The objective comparisons with methods BM3D [4], VBM3D [25], BM4D [26], VBM4D [27] in terms of the PSNR and the SSIM on the 'carpark' example. We vary the noise levels. The performance of our method is on par with the other state-of-the-art methods.

5. CONCLUSION

We have presented a novel method that enables video coding for video denoising. With the extracted Coding Trajectory from the video coding, our method can achieve high quality results while maintaining a high efficiency. Various cases demonstrate the robustness and the effectiveness of our proposed method.

6. ACKNOWLEDGE

This work has been supported by National Natural Science Foundation of China (61502079, 61672134 and 61720106004).

References

[1] P. Chatterjee, N. Joshi, S. Kang, and Y. Matsushita, "Noise suppression in low-light images through joint denoising and demosaicing," in *Proc. CVPR*, 2011, pp. 321–328.

[2] J. Chen, C. Tang, and J. Wang, "Noise brush: interactive high quality image-noise separation," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 146, 2009.

[3] X. Chen, B. Sing, J. Yang, and J. Yu, "Fast patch-based denoising using approximated patch geodesic paths," in *Proc. CVPR*, 2013, pp. 1211–1218.

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

[5] Z. Liu, L. Yuan, X. Tang, M. Uyttendaele, and J. Sun, "Fast burst images denoising," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 232, 2014.

[6] Z. Ren, J. Li, S. Liu, and B. Zeng, "Meshflow video denoising," in *Proc. ICIP*, 2017.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "Surf: Speeded up robust features," in *Proc. ECCV*, 2006, pp. 404–417.

[8] S. Liu, P. Tan, L. Yuan, J. Sun, and B. Zeng, "Meshflow: Minimum latency online video stabilization," in *Proc. ECCV*, 2016, pp. 800–815.

[9] S. Cho, J. Wang, and S. Lee, "Vdeo deblurring for handheld cameras using patch-based synthesis," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 64:1–64:9, 2012.

[10] M. Granados, K. Kim, J. Tompkin, and C. Theobalt, "Automatic noise modeling for ghost-free hdr reconstruction," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 201, 2013.

[11] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 78, 2013.

[12] C. Barnes, F. Zhang, L. Lou, X. Wu, and S. Hu, "Patchtable: efficient patch queries for large datasets and applications," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 97, 2015.

[13] F. Tan, S. Liu, L. Zeng, and B. Zeng, "Kernel-free video deblurring via synthesis," in *Proc. ICIP*, 2016, pp. 2683–2687.

[14] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Proc. CVPR*, 2005, vol. 2, pp. 60–65.

[15] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[16] B. Zeng and J. Fu, "Directional discrete cosine transformsła new framework for image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 3, pp. 305–313, 2008.

[17] S. Zhu, S. Yeung, and B. Zeng, "In search of better-than-dct unitary transforms for encoding of residual signals," *IEEE Signal Processing Letters*, vol. 17, no. 11, pp. 961–964, 2010.

[18] P. Meher, S. Park, B. Mohanty, K. Lim, and C. Yeo, "Efficient integer dct architectures for hevc," *IEEE Transactions on Circuits and systems for Video Technology*, vol. 24, no. 1, pp. 168–178, 2014.

[19] Y. Taki, M. Hatori, and S. Tanaka, "Interframe coding that follows the motion," *Proc. Institute of Electronics and Communication Engineers Jpn. Annu. Conv.(IECEJ)*, p. 1263, 1974.

[20] T. Koga, "Motion-compensated interframe coding for video conferencing," in *proc. NTC 81*, 1981, pp. C9–6.

[21] L. Po and W. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE transactions on circuits and systems for video technology*, vol. 6, no. 3, pp. 313–317, 1996.

[22] S. Zhu and K. Ma, "A new diamond search algorithm for fast block matching motion estimation," in *Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on*. IEEE, 1997, vol. 1, pp. 292–296.

[23] E. Balster, Y. Zheng, and R. Ewing, "Combined spatial and temporal domain wavelet shrinkage algorithm for video denoising," *IEEE Trans. on Circuits and Syst. for Video Tech.*, vol. 16, no. 2, pp. 220–230, 2006.

[24] N. Kalantari, E. Shechtman, C. Barnes, S. Darabi, D. B Goldman, and P. Sen, "Patch-based high dynamic range video.," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 202, 2013.

[25] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3d transform-domain collaborative filtering," in *In Proc. European Signal Process. Conf. EUSIPCO*, 2007.

[26] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE Trans. on Image Processing*, vol. 22, no. 1, pp. 119–133, 2013.

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

[28] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.