# Parallelized and Vectorized Implementation of DCT denoising with FMA instructions

# Parallelized and Vectorized Implementation of DCT denoising with FMA instructions

Yuki Kawasaki, Yoshihiro Maeda, and Norishige Fukushima*
Nagoya Institute of Technology, Japan
Email: *fukushima@nitech.ac.jp

***Abstract*—DCT denoising is a denoising technique, which filtering an image in the frequency domain. The DCT denoising is known as an excellent method in a balance between processing time and denoising accuracy. In this paper, we implement the DCT denoising for further improving the computational cost to use efficient implementation of DCT, named AAN (Arai-Agui-Nakajima's methods). Also, we utilize FMA (fused multiply-add) instructions for the AAN-based DCT for accelerations. In the experiments, we compare the proposed DCT algorithm with DCT denoising based on Chen's algorithm, which is a normal fast DCT algorithm. The experimental results show that the proposed method is superior to the conventional method.**

***Keywords*—DCT Denoising, AAN, FMA, SIMD**

## I. Introduction

Denoising is an essential process in image processing, and there are enormous researches on this topics. The denoising methods are roughly classified into a filtering in a spatial domain and a frequency domain.

Represent methods in the spatial domain filtering are bilateral filtering [1], non-local means filtering [2], and guided image filtering [3]. These filters are FIR convolution filters, and these computational costs depend on filtering kernel size. Therefore, there are several acceleration algorithms in bilateral filtering [4], [5], [6], [7], [8], [9], [10], [11], [12], non-local means filtering [13], [11], and guided image filtering [14], [15]. Each acceleration is aimed at approximating brute-force implementation; hence denoising performance tends to be reduced. Furthermore, frequency domain filters have superior denoising performance to the spatial domain filtering.

Represent methods in frequency domain filtering are wavelet shrinkage [16], [17], BM3D [18], and DCT denoising [19]. The wavelet shrinkage denoising is an early method, which utilizes partial frequency domain transform. Partial frequency transforms extract more detail features in an image than transforming whole image signals. Denoising with Bayesian least squares based on Gaussian scale mixture (BLS-GSM) [17] is an early method, which utilizes redundant frequency transform. BM3D is the state-of-the-art method in denoising performance, which use redundant frequency transform with 3D signals; however, its computational cost is high. DCT denoising is a simplified method of the BM3D and is an excellent method of balancing calculation speed and performance. Therefore the DCT denoising is utilized in FFMPEG, which is de-facto standard video processing software.

In this paper, we propose an accelerated method for the DCT denoising to improve the computational performance further. Randomized redundant DCT denoising algorithm accelerate DCT denoising by subsampling the number of DCT patches, but the proposed method does not use subsampling, thence the proposed method keep accuracy from the brute-force implementation. We utilize a fast implementation of DCT, named AAN [20], which is used for DCT conversion in the JPEG image compression algorithm. Furthermore, we accelerate the AAN with the FMA intrinsic, which is vector operation of simultaneous multiplication and addition.

## II. DCT denoising

In DCT denoising, an image is delimited by redundant blocks, and the blocked patches are denoised. A block is transformed into a frequency domain by using DCT, and then small DCT coefficients in the patch are thresholded. Next, the patch is inversely transformed. We iterate these processes for the whole patch. Finally, the overlapped pixels in patches are averaged to compose the entire image.

Figure 1 shows the detail processing flow of DCT denoising. $P$ represents a set of patches that be converted by DCT. First, we convert the $i$-th patch $f_i$ in $P$ to the frequency domain by DCT.

$$F_i = S^{DCT}(f_i), \tag{1}$$

where $F_i$ represents DCT coefficients of the patch $f_i$, and $S^{DCT}(\cdot)$ represents a DCT function. Next, when absolute values of the DCT coefficients are smaller than a threshold value, the coefficients are regarded as noisy signals, and then are replaced with 0, which is called hard thresholding. The process is as follow;

$$F_i'(u,v) = \begin{cases} F_i(u,v) & u = v = 0 \\ F_i(u,v) & |F_i(u,v)| > th \\ 0 & otherwise, \end{cases} \tag{2}$$

where $F_i'$ is coefficients after thresholding and $th$ is the threshold. Note that the DC component should be held; thus thresholding is not performed for DC. Finally, $F_i'$ is inversely transformed into the spatial domain by IDCT (Inverse DCT).

$$f_i' = S^{IDCT}(F_i'), \tag{3}$$

where $f_i'$ represents the denoised patch, and $S^{IDCT}(\cdot)$ represents the IDCT function. This process is done for all patches

Fig. 1: Algorithm of DCT denoising.

in $P$. Finally, it outputs by averaging pixels with overlapping patches.

$$J_p = \sum_{i \in \omega_p} f_i'(q_p), \tag{4}$$

where $\omega_p$ is a patch set, which include the pixel $p$, and $q_p$ indicates that a coordinate of the i-th patch $q_i = s, t = p$. The patches are averaged in the same weight; thus this per patch operations can be merged into whole image processing.

$$J = 1/N \sum_{i \in \Omega} f_i', \tag{5}$$

where N is the size of the patch, e.g., in the $8 \times 8$ case, N=64.

## III. Fast DCT by AAN

Fast implementation of DCT, named AAN, will be described here. For simple calculation of 8-point DCT, this conversion requires multiplying a vector, which has 8 elements, by a transformation matrix of $8 \times 8$. In this case, the required calculation is 64 multiplications and 56 additions. The 2D case has 512 multiplications and 448 additions.

In Chen's algorithm, which is a basic butterfly computation for DCT, 1D DCT requires 18 multiplications and 28 additions. Also, 2D DCT requires 288 multiplications and 448 additions.

On the other hand, AAN [20] can perform 8-points DCT calculation, 5 multiplications and 29 additions, and 8 multiplications for scaling. Figure 2 shows the diagram of AAN. In this figure, it is shown that the sum of squares at the intersection points and the product of the coefficients in the squares are obtained. Also, a black triangle indicates sign inversion. Note that 2D DCT with AAN, such as $8 \times 8$ DCT, requires two times scaling, but this scaling can be merged and simultaneously performed at once. Therefore, the number of calculations of $8 \times 8$ DCT in this implementation has 144 multiplications, which includes 64 scaling and 80 multiplications, and 464 additions. IDCT can be realized by calculating the diagram from the inverse, and the number of calculations is the same as in the case of DCT. In IDCT, scaling is done before butterfly computation.

Fig. 2: Diagram of AAN.

Fig. 3: Forward DCT of AAN with FMA instructions.

## IV. Proposed methods

### A. AAN for DCT denoising

In the AAN butterfly, scaling processing comes after the final step of the DCT computation or comes before the first one of the IDCT butterfly. By performing the thresholding and the scaling collectively before and after the DCT butterfly, the number of calculations can be reduced. More specifically, when thresholding and scaling are performed only for coefficients whose absolute value exceeds the threshold $t(u, v)$. The threshold $t(u, v)$ is obtained by AAN scaling factor by dividing the threshold $th$. The proposed DCT process is as follows;

$$F_i'(u,v) = \begin{cases} F_i(u,v) \times M(u,v) & u = v = 0 \\ F_i(u,v) \times M(u,v) & |F_i(u,v)| > t(u,v) \\ 0 & otherwise, \end{cases} \tag{6}$$

$$M(u,v) = (m_u \times m_v) \times (m_u \times m_v) \tag{7}$$

$$t(u,v) = \frac{th}{m_u \times m_v} \tag{8}$$

where $m_u, m_v$ are the scaling values of one-dimensional DCT in each of the vertical and horizontal directions, and $M(u, v)$ is the scaling value of the two-dimensional DCT. Figure 5 shows the flow of the conventional DCT denoising and the proposed

Fig. 4: Inverse DCT of AAN with FMA instructions.

Fig. 5: AAN for DCT denoising.

one. Merging the dual scaling and thresholding processes, the number of multiplications is reduced. Note that when the coefficient is replaced with 0, the number of multiplications is further reduced by not performing scaling.

### *B. DCT denoising with FMA instructions*

The FMA can compute $(A \times B) + C$ with one operation.

In the fig. 3 shows AAN with FMA, where enclosed region indicates multiplication and addition are combined by FMA.

In the inverse DCT case, more FMA operations are used, since the scaling in the first command of IDCT can be integrated. Therefore, Eight times of integration required for scaling can be reduced to 4 times by using FMA instructions. In the fig. 4, a dotted line indicate the area where the number of instructions can be reduced with the FMA instruction in AAN IDCT. In this figure, it is shown that the sum of squares at the intersection points and the product of the coefficients in the squares are obtained. Also, a black triangle indicates sign inversion.

The number of operations for the computation of patches of $8 \times 8$ is 136 times when using Chen's algorithm, and 124 times using the proposed method.

## V. Experimental Results

In the experiments, we compared our implementation with the conventional implementation. We used an image whose size was $512 \times 512$ pixels as an input image. We compared the AAN based-DCT denoising with Chen's algorithm-based one. We also contrasted the methods implemented with/without the FMA instruction for each method. In this experiment, we used Core i7-6700K 4.0 GHz (4 cores 8 threads) with Visual Studio 2015 C++ compiler.

Fig. 6: Result image of DCT denoising.

TABLE I: Computational time (C++).

| | Chen | AAN |
|---|---|---|
| time [msec] | 924.41 | 865.61 |

TABLE II: Computational time (parallelized and vectorized).

| | Chen | AAN | Chen - FMA | AAN - FMA |
|---|---|---|---|---|
| time [msec] | 23.12 | 22.64 | 19.61 | 16.57 |

TABLE I shows the processing time of each method, and TABLE II shows the processing time of each method with vectorization and parallelization. "- FMA" in the table indicates that the implementations utilize the FMA. The result shows that our method is faster than the conventional method in each case.

Figure 6 shows the example of denoising results. The result image indicates both conventional and proposed images since the proposed method has the almost same quality of the brute-force implementation expecting for floating value computation error.

Table III shows denoising result of fast denoising methods. The proposed method has the best denoising performance. Figure 7 shows computational time of each denoising method with the same parameter setting in Table III. The proposed method is the second best method. The DCT denoising has also implemented in OpenCV library, but the proposed method quite faster than this implementation. In Table IV, PSNR and SSIM of each denoising method for 24 color images have been provided by Kodak corporation. The proposed method has the best denoising performance.

## VI. Conclusion

In this paper, we proposed an acceleration implementation of DCT denoising by using the AAN DCT and FMA operation. Experimental results showed that the proposed implementation improved the computational speed of the DCT denoising without loss of accuracy. Also, the fast implementation of the DCT denoising was 13.33 ms for $512 \times 512$ images, which was enough performance for 60 fps video.

The acceleration of DCT operation could further apply DCT based denoising, e.g., BM3D, two-step DCT denoising, these examples also could be improved.

## Acknowledgement

This work was supported by KAKENHI JP17H01764.

TABLE III: PSNR of each denoising method. BF is bilateral filtering, NLM is non-local means.

| | BF | NLM | proposed |
|---|---|---|---|
| PSNR [db] | 31.90 | 30.12 | 32.12 |

Fig. 7: Computational time of each denoising method. Image size is $512 \times 512$.

TABLE IV: PSNR and SSIM of each denoising method for Kodak Image. BF is bilateral filtering, NLM is non-local means.

| | | BF | NLM | proposed |
|---|---|---|---|---|
| kodim01 | PSNR [db] | 25.59 | 27.82 | 29.69 |
| | SSIM | 0.71 | 0.80 | 0.86 |
| kodim02 | PSNR [db] | 26.84 | 31.33 | 32.41 |
| | SSIM | 0.52 | 0.77 | 0.82 |
| kodim03 | PSNR [db] | 26.80 | 32.28 | 33.82 |
| | SSIM | 0.49 | 0.81 | 0.89 |
| kodim04 | PSNR [db] | 26.53 | 31.25 | 32.31 |
| | SSIM | 0.53 | 0.79 | 0.84 |
| kodim05 | PSNR [db] | 25.81 | 28.37 | 29.84 |
| | SSIM | 0.72 | 0.84 | 0.88 |
| kodim06 | PSNR [db] | 26.08 | 28.79 | 30.64 |
| | SSIM | 0.64 | 0.79 | 0.86 |
| kodim07 | PSNR [db] | 26.44 | 31.65 | 33.34 |
| | SSIM | 0.57 | 0.86 | 0.92 |
| kodim08 | PSNR [db] | 25.61 | 27.75 | 29.71 |
| | SSIM | 0.76 | 0.85 | 0.89 |
| kodim09 | PSNR [db] | 26.52 | 31.71 | 33.99 |
| | SSIM | 0.51 | 0.82 | 0.90 |
| kodim10 | PSNR [db] | 26.54 | 31.58 | 33.41 |
| | SSIM | 0.52 | 0.81 | 0.88 |
| kodim11 | PSNR [db] | 26.37 | 29.78 | 31.33 |
| | SSIM | 0.59 | 0.78 | 0.84 |
| kodim12 | PSNR [db] | 26.68 | 31.63 | 33.31 |
| | SSIM | 0.50 | 0.78 | 0.86 |
| kodim13 | PSNR [db] | 25.36 | 26.49 | 28.20 |
| | SSIM | 0.75 | 0.77 | 0.83 |
| kodim14 | PSNR [db] | 26.03 | 29.17 | 30.31 |
| | SSIM | 0.64 | 0..78 | 0.83 |
| kodim15 | PSNR [db] | 27.12 | 31.17 | 32.13 |
| | SSIM | 0.56 | 0.81 | 0.86 |
| kodim16 | PSNR [db] | 26.36 | 30.51 | 32.43 |
| | SSIM | 0.54 | 0.77 | 0.86 |
| kodim17 | PSNR [db] | 26.89 | 31.44 | 32.77 |
| | SSIM | 0.58 | 0.83 | 0.88 |
| kodim18 | PSNR [db] | 26.13 | 28.79 | 30.11 |
| | SSIM | 0.63 | 0.78 | 0.84 |
| kodim19 | PSNR [db] | 26.27 | 30.09 | 31.75 |
| | SSIM | 0.58 | 0.79 | 0.85 |
| kodim20 | PSNR [db] | 27.77 | 30.19 | 30.91 |
| | SSIM | 0.69 | 0.86 | 0.89 |
| kodim21 | PSNR [db] | 26.16 | 29.48 | 31.31 |
| | SSIM | 0.58 | 0.81 | 0.88 |
| kodim22 | PSNR [db] | 26.24 | 29.94 | 31.10 |
| | SSIM | 0.56 | 0.76 | 0.82 |
| kodim23 | PSNR [db] | 26.89 | 32.74 | 34.62 |
| | SSIM | 0.48 | 0.83 | 0.90 |
| kodim24 | PSNR [db] | 26.06 | 28.76 | 30.09 |
| | SSIM | 0.64 | 0.81 | 0.87 |

## REFERENCES


[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *IEEE International Conference on Computer Vision (ICCV)*, 1998, pp. 839–846.

[2] A. uades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 60–65.

[3] K. He, J. Shun, and X. Tang, "Guided image filtering," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 1–14.

[4] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions. on Graphics*, vol. 21, no. 3, pp. 257–266, 2002.

[5] T. Q. Pham and L. J. V. Vliet, "Separable bilateral filtering for fast video preprocessing," in *in Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2005, pp. 1–4.

[6] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 568–580.

[7] F. Porikli, "Constant time o(1) bilateral filtering," in *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[8] Q. Yang, K. H. Tan, and N. Ahuja, "Real-time o(1) bilateral filtering," in *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 557–564.

[9] K. Chaudhury, "Constant-time filtering using shiftable kernels," *IEEE Signal Processing Letters*, vol. 18, no. 11, pp. 651–654, 2011.

[10] ——, "Acceleration of the shiftable mbiO(1) algorithm for bilateral filtering and nonlocal means," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1291–1300, 2013.

[11] N. Fukushima, S. Fujita, and Y. Ishibashi, "Switching dual kernels for separable edge-preserving filtering," in *in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 1588–1592.

[12] K. Sugimoto and S.-I. Kamata, "Compressive bilateral filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3357–3369, 2015.

[13] Y. S. Kim, H. L., O. Choi, K. Lee, J. Kim, and J. Kim, "Separable bilateral nonlocal means," in *IEEE International Conference on Image Processing (ICIP)*, 2011, pp. 1513–1516.

[14] S. Fujita and N. Fukushima, "High-dimensional guided image filtering," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2016, pp. 1–8.

[15] ——, *Extending Guided Image Filtering for High-Dimensional Signals*. Springer International Publishing, 2017, vol. 693, pp. 439–453.

[16] S. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1532–1546, 2000.

[17] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixture of gaussians in the wavelet domain," *in Proceedings of the IEEE Transactions on Image Processing*, vol. 12, pp. 1338–1351, 2003.

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

[19] G. Yu and G. Sapiro, "Dct image denoising: A simple and effective image denoising algorithm," *Image Processing On Line*, vol. 1, p. 1, 2011.

[20] Y. Arai, T. Agui, and M. Nakajima, "A fast dct-sq scheme for images," *Transactions on IEICE*, vol. E-71, no. 11, pp. 1095–1097, 1988.