Randomized Redundant DCT:
Efficient Denoising by Using Random Subsampling of DCT Patches

Shu Fujita[1] Norishige Fukushima[*1] Makoto Kimura[2] Yutaka Ishibashi[1]
[1]Nagoya Institute of Technology, Japan
[2]Samsung R&D Institute Japan

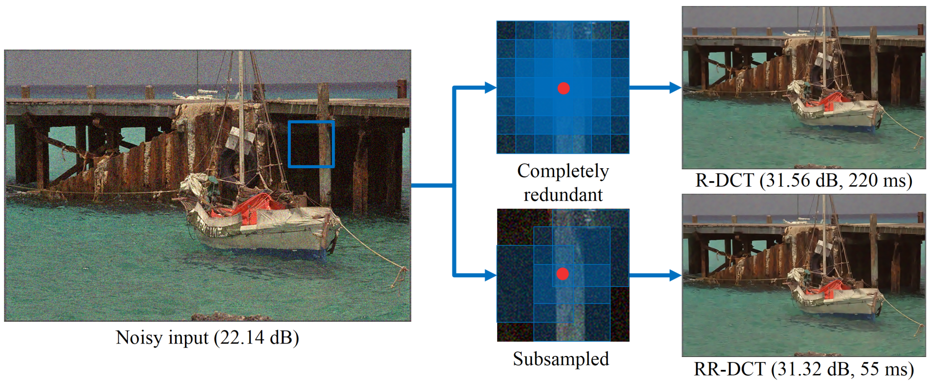


Figure 1: *Overview of proposed and conventional denoising methods. The conventional method utilizes completely redundant patches; while our method performs random subsampling for the patches. The random subsampling can accelerate the denoising method, while the denoising performance is preserved.*

Abstract

In this paper, we propose an acceleration method for image denoising with a redundant discrete cosine transform (R-DCT). Image denoising is essential for image processing, and its efficiency is important for graphics applications. R-DCT with a hard-thresholding or shrinkage method can perform denoising while keeping detail textures. Moreover, the method is computationally efficient compared with state-of-the-art denoising methods, such as BM3D. The computational cost, however, is still insufficient for real-time processing; hence, we accelerate the method by using randomized subsampling of DCT patches. Experimental results show that our method can accelerate the processing while the degradation of denoising performance is a little.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement—Smoothing

Keywords: image denoising, redundant DCT, real-time denoising, randomized sampling

1 Introduction

Image denoising is a fundamental issue in image processing, and it is often required as pre-processing or post-processing in graphics applications. For real-time applications, the efficiency of image denoising is essential. Especially in computer graphics applications, we should focus the trade-off between denoising performance and its efficiency.

Among image denoising methods, patch-based methods achieve high performance. Non-local means filtering [Buades et al. 2005] is a representative of spatial domain denoising and has several acceleration methods [Adams et al. 2009; Adams et al. 2010; Gastal and Oliveira 2012; Fukushima et al. 2015]. Detailed textures, however, tend to be over-smoothed using spatial domain approaches. As the other patch-based method, BM3D [Dabov et al. 2007] is one of the state-of-the-art methods [Knaus and Zwicker 2014]. BM3D has two steps: spatial domain processing and frequency domain processing. BM3D has excellent denoising performance while the method is computationally expensive.

Denoising based *redundant discrete cosine transform (R-DCT)* [Yu and Sapiro 2011], which is a patch-based and frequency domain approach, has a fair trade-off between denoising performance and computational efficiency. Thus, R-DCT is implemented in FFMPEG of famous video encoder/decoder. R-DCT denoising performs a thresholding or shrinkage process for each patch in the frequency domain as the other frequency domain approaches [Chang et al. 2000; Starck et al. 2002]. Then, the processing is performed redundantly by using overlapping patches. The method has better denoising performance than the non-local means filtering, and the computational cost less than the non-local means filtering. The computational cost, however, is slightly insufficient for real-time applications.

Therefore, we propose an acceleration method for denoising based R-DCT in this paper. The redundancy is important for denoising performance in the R-DCT denoising; however, the improvement of the denoising performance from the redundancy is saturated. Based on the over-redundancy, we subsample the redundant patches. Moreover, we introduce a randomized subsampling method—named as *randomized redundant DCT (RR-DCT)*—for aliasing issues, which is inevitable in subsampling. The accelerated code for RR-DCT is downloadable [1].

*http://fukushima.web.nitech.ac.jp/en/

[1]http://fukushima.web.nitech.ac.jp/research/rrdct/

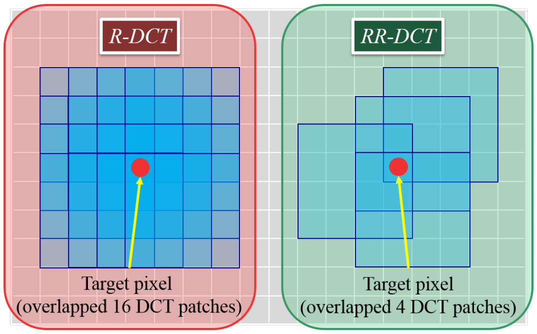


Figure 2: *Degree of overlapping of R-DCT and RR-DCT patches around a target pixel. The patch size is* 4×4.

2 Redundant DCT

We review the R-DCT denoising proposed by Yu and Sapiro [Yu and Sapiro 2011] in this section. Let Ω be a set of all patches of size $N \times N$ in an input image I. R-DCT denoising firstly transforms the i-th patch $f_i \in \Omega$ into signals in the frequency domain:

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

where $\psi^{\mathrm{DCT}}(\cdot)$ represents a forward DCT function, and F_i has DCT coefficients of the patch f_i. We regard DCT coefficients as noises if the DCT coefficients are less than a threshold value τ. Then, we discard them by hard-thresholding:

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

where F_i' is the coefficients after hard-thresholding. Note that the direct-current (DC) component must be protected, i.e., the F_i' always satisfies that $F_i'(0,0) = F_i(0,0)$; thus, we perform the hard-thresholding without the DC component. The refined coefficients F_i' are re-transformed into spatial domain signals:

$$f_i' = \psi^{\mathrm{iDCT}}(F_i'), \tag{3}$$

where f_i' is a denoised patch, and $\psi^{\mathrm{iDCT}}(\cdot)$ represents an inverse DCT function. R-DCT denoising performs these processes for each patch in the set Ω.

Each patch includes overlapping areas as shown in the left-side of Fig. 2. R-DCT denoising averages the all overlapped patches, that is, the value of the output image at pixel $\boldsymbol{p}$ is as follows:

$$I'(\boldsymbol{p}) = \frac{1}{|\omega(\boldsymbol{p})|} \sum_{f_i' \in \omega(\boldsymbol{p})} f_i'(\mathrm{map}_i(\boldsymbol{p})), \tag{4}$$

where $\omega(\boldsymbol{p}) \subset \Omega$ represents a set of patches including a pixel $\boldsymbol{p}$, and $|\omega(\boldsymbol{p})|$ represents the number of elements of $\omega(\boldsymbol{p})$. $\mathrm{map}_i(\cdot)$ represents a mapping function of corresponding positions between the focusing pixel $\boldsymbol{p}$ and the corresponding position in the i-th patch.

For color image processing, the method performs a color decorrelation for RGB channels as pre-processing and then perform denoising for each channel. In this color decorrelation, we compute 3 point DCT for color vectors and use the following orthonormal basis: $\{(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})^T, (\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}})^T, (\frac{1}{\sqrt{6}}, -\frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}})^T\}$. The color decorrelation can improve the denoising performance than a standard color transformation, such as the YUV and Lab color space.

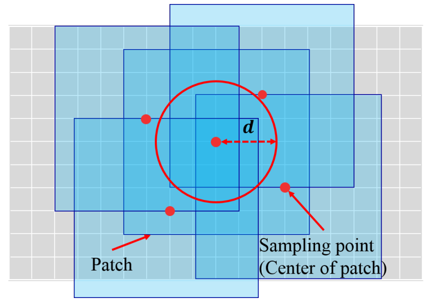
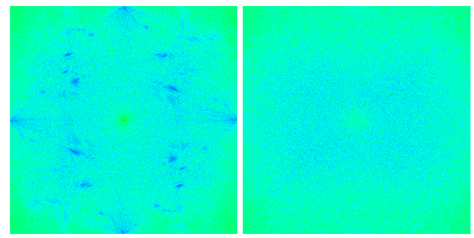


Figure 3: *Example of Poisson-disk subsampling for patches. This sampling method can uniformly sample patches.*



(a) *RS* **(b)** *PDS*

Figure 4: *Frequency domain of differential signal between subsampled and full sampling signal. The input image is "barbara".*

3 Randomized Redundant-DCT

We introduce our method of RR-DCT. Here, we have one patch per one pixel in the case of the full-patch set; however, the patches have much redundancy. Thus, we subsample the patches. We use a subset Ω_s of the full-patch set Ω. An example of subsampled patches is shown in the right-side of Fig. 2. Letting $\omega_s(\boldsymbol{p}) \subset \Omega_s$ be a subset of the subsampled patches including a pixel $\boldsymbol{p}$, Eq. (4) is re-written as:

$$I'(\boldsymbol{p}) = \frac{1}{|\omega_s(\boldsymbol{p})|} \sum_{f_i' \in \omega_s(\boldsymbol{p})} f_i'(\mathrm{map}_i(\boldsymbol{p})). \tag{5}$$

The subsampling method is important for denoising performance. The intuitive subsampling is dropping off the patches at a regular interval—named regular subsampling (RS). In the extreme RS case, patches have no overlap like DCT processing of JPEG image coding. Unfortunately, RS causes aliasing in reconstructed images; hence, we introduce a randomized subsampling method for preventing the degradation.

If we perform purely random sampling, the density of sampling has not equality. In the worst case, these are no patches for a denoising pixel. Here, Poisson-disk sampling (PDS) [Cook 1986; Dunbar and Humphreys 2006] has a suitable property for the random subsampling [Banterle et al. 2012]. The subsampling method selects points that the distance of the points among the samples should be d at least. For extending the patch subsampling, we drop-off the patches, which have the defined distance from the center of the nearest patches. The defined distance d should be shorter than the patch width/height. The example of sampling is shown in Fig. 3.

We show the frequency domain signals of the differential signals between subsampled and full sampled signals in Fig. 4. The subsampling methods are RS and PDS. Note that the sampling densities are almost the same. We can see the biases in frequency characteristics of RS while PDS has almost flat characteristics.

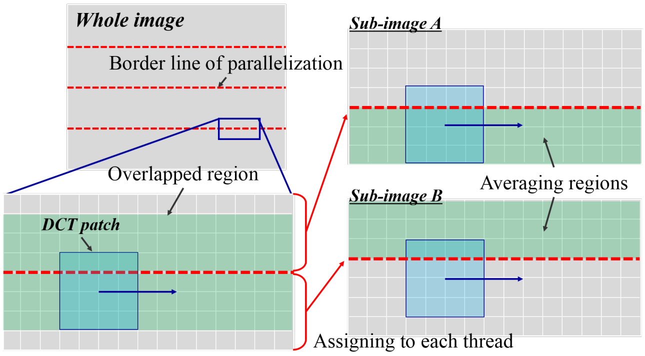


Figure 5: *Diagram of implementation in parallelization.*

4 Implementation

We present the implementation of RR-DCT. For efficient implementation, we discuss two points: Look-up-table (LUT) for randomized sampling patterns and parallelization of our denoising.

At first, we prepare several patterns of PDS to LUTs before denoising for saving the computational cost of generating patterns. We have to prepare a limited number of LUTs for randomness and switch the LUTs image by image.

The next point is parallelization. The proposed denoising method contains three steps; patch-sampling (Sec. 3), patch-denoising (Eq. (1)-(3)) and patch-averaging (Eq. (4) or (5)). The patch-sampling and patch-denoising can be massively parallelized; however, patch-averaging is a typical reduction pattern in parallel processing [McCool et al. 2012]. Patches are the minimum processing unit for this denoising; thus all process must be in the reduction pattern, which is not effective than the fully parallelized case.

For avoiding the reduction pattern, we divide an input image into multiple sub-images, which have the copy image of the overlapped regions (See Fig. 5). For each sub-image, we perform patch-sampling, patch-denoising and counting the number of patches. Note that we parallelly process these processes par sub-image. Finally, we parallelly average the overlapped region of sub-images by using reduction pattern. The averaging process is the lowest cost step; thus we can save the degradation of performance from the reduction process.

This parallelization is suitable for CPUs, which do not have hundreds of cores. Note that the process is not scale to massively multi-core devices, such as GPU. The parallelization is efficient for a small number of cores. To moderate the overhead, additonal vertical separations for sub-images is useful for many-core cases.

5 Experimental Results

We verify the accuracy and efficiency in this section. For accuracy evaluations, we use PSNR. We use 8×8 DCT patch and set the thresholding value τ to the same value of noise level σ. We implement LLM-based-DCT with SIMD vectorization for fast computing [Loeffler et al. 1989]. For the other patch size case, we utilize Plonka's DCT factorization[Plonka and Tasche 2005].

Accuracy performance

We firstly show the results of the denoising accuracy of our method. The comparison results between the brute-force implementation [Yu and Sapiro 2011] and ours are shown in Fig. 6 and 7. Note that we show the cases of $d = 2, 3$ where is the minimum distance for PDS. The subsampling rates of the patches are about 20% and 8%, respectively.

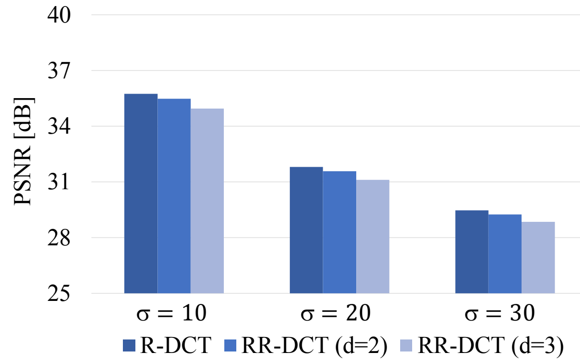
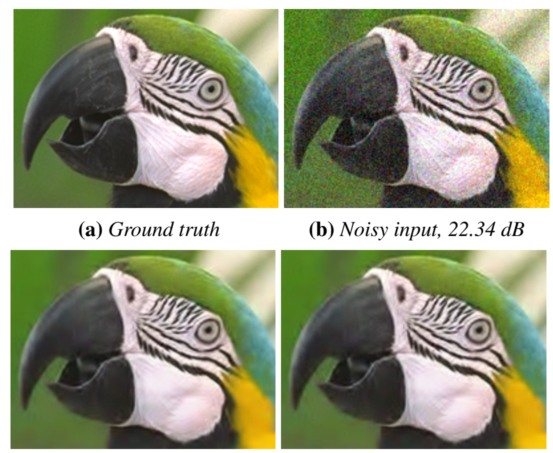


Figure 6: *Comparison of PSNR accuracy. We use Kodak PhotoCD dataset (**24 images**), and this result is the average value. Note that the average PSNR of each input image are 28.23 dB, 22.33 dB, and 18.98 dB, respectively. The standard deviation of RR-DCT is about* 6.1×10^{-3}*, which the number of trials is 1000.*



(a) *Ground truth* **(b)** *Noisy input, 22.34 dB*

(c) *R-DCT, 34.37 dB* **(d)** *RR-DCT(d=2), 34.11 dB*

Figure 7: *Image denoising results by R-DCT. (a) Exact image. (b) Noisy image added with Gaussian noises where the standard deviation* $\sigma = 20$*. (c) Full sampled result (R-DCT). (d) Our result* $(d = 2)$*.*

Figure 6 shows the denoising performance for various images. We use *24 images* in Kodak PhotoCD dataset. The results represent that the degradation of denoising performances is low even if we drop-off many patches. Furthermore, the variation of PSNR is quite small. The fact is also visually shown in Fig. 7.

Computational performance

We demonstrate the computational performance in this experiment. The resolution of the test image is 1024×1024 (*one megapixel*). We have vectorized the C++ code for the proposed and competitive methods by using SIMD intrinsics with Visual Studio 2010. The CPU is Intel Core i7-3770K 3.50 GHz on Windows 7 64 bit.

Figure 8 shows the result of the computational performance. Our method becomes faster as the subsampling rate increases. Moreover, we can accelerate $\times 2$ or $\times 3$ by parallelizing in our experimental environment with 4 core CPU.

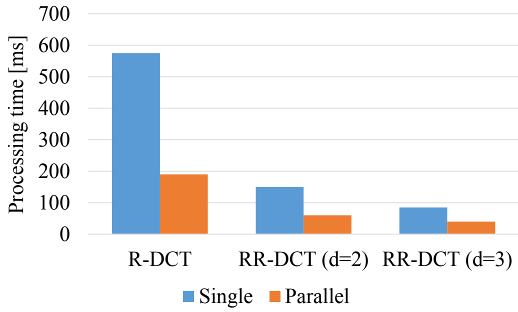


Figure 8: *Processing time with a one megapixel color image.*

(a) *Ground truth* **(b)** *Noisy input* **(c)** *BM3D, 4100 ms*

(d) *AM, 140 ms* **(e)** *R-DCT, 72 ms* **(f)** *RR-DCT, 23 ms*

Figure 9: *Denoising results by various methods. (a) Ground truth. (b) Noisy input image added Gaussian noises where the standard deviation* $\sigma = 20$*. (c) BM3D result. (d) AM (6-D) result. (e) R-DCT result. (d) Our result (*$d = 2$*). The image size is* 768×512*.*

Comparison with state-of-the-art methods

We compare our RR-DCT denoising with the state-of-the-art denoising methods. We use BM3D [Dabov et al. 2007] and adaptive manifolds (AM) [Gastal and Oliveira 2012] as our competitive methods. Here, BM3D is the best denoising performance method, and we implement BM3D by C++ with FFTW. AM is one of the real-time denoising methods, which is the fastest acceleration of non-local means filtering (but AM is not limited to denoising applications). We use OpenCV's implementation for AM.

Figure 9 shows the comparison of them. Although AM oversmooths the image, our result can preserve detail textures. BM3D has the best denoising performance; however it takes $\times 200$ processing time compared to ours.

6 Conclusions

We have proposed an acceleration method for image denoising with redundant DCT. The acceleration is achieved by randomized subsampling of patches—named as randomized redundant DCT (RR-DCT). The randomized subsampling supports denoising performance and computational efficiency at the same time. As a result, ours has the best real-time denoising performance among the competitive methods.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 15K16023.

References

Adams, A., Gelfand, N., Dolson, J., and Levoy, M. 2009. Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. on Graphics 28*, 3.

Adams, A., Baek, J., and Davis, M. A. 2010. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum 29*, 2, 753–762.

Banterle, F., Corsini, M., Cignoni, P., and Scopigno, R. 2012. A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain. *Computer Graphics Forum 31*, 1, 19–32.

Buades, A., Coll, B., and Morel, J. M. 2005. A non-local algorithm for image denoising. In *Proc. CVPR*, 60–65.

Chang, S., Yu, B., and Vetterli, M. 2000. Adaptive wavelet thresholding for image denoising and compression. *IEEE Trans. on Image Processing 9*, 9, 1532–1546.

Cook, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. on Graphics 5*, 1, 51–72.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. 2007. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. on Image Processing 16*, 8, 2080–2095.

Dunbar, D., and Humphreys, G. 2006. A spatial data structure for fast poisson-disk sample generation. *ACM Trans. on Graphics 25*, 3, 503–508.

Fukushima, N., Fujita, S., and Ishibashi, Y. 2015. Switching dual kernels for separable edge-preserving filtering. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Gastal, E. S. L., and Oliveira, M. M. 2012. Adaptive manifolds for real-time high-dimensional filtering. *ACM Trans. on Graphics 31*, 4.

Knaus, C., and Zwicker, M. 2014. Progressive image denoising. *IEEE Trans. on Image Processing 23*, 7, 3114–3125.

Loeffler, C., Ligtenberg, A., and Moschytz, G. S. 1989. Practical fast 1-d dct algorithms with 11 multiplications. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 988–991.

McCool, M., Reinders, J., and Robison, A. 2012. *Structured parallel programming: patterns for efficient computation*. Elsevier.

Plonka, G., and Tasche, M. 2005. Fast and numerically stable algorithms for discrete cosine transforms. *Linear algebra and its applications 394*, 309–345.

Starck, J.-L., Candes, E., and Donoho, D. 2002. The curvelet transform for image denoising. *IEEE Trans. on Image Processing 11*, 6, 670–684.

Yu, G., and Sapiro, G. 2011. Dct image denoising: A simple and effective image denoising algorithm. *Image Processing On-Line 1*.