IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 17, 2021, accepted June 30, 2021, date of publication July 12, 2021, date of current version July 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3096521

Similar Patch Selection in Embedding Space for Multi-View Image Denoising

GEUNWOO OH[1], (Student Member, IEEE), DONG-WAN CHOI[2], (Member, IEEE), AND BOCHANG MOON[1], (Member, IEEE)

[1]School of Integrated Technology, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea
[2]Department of Computer Science and Engineering, Inha University, Incheon 22212, South Korea

Corresponding author: Bochang Moon (bmoon@gist.ac.kr)

This work was supported by the Ministry of Culture, Sports and Tourism and Korea Creative Content Agency under Project R2020070004.

ABSTRACT This paper proposes an image patch selection that finds similar patches in multiple images so that image denoising can suppress noise more effectively by exploiting the identified similar patches from the multi-view images. We encode all image patches in multi-view images into a low-dimensional space, and it allows for a denoiser to find similar patches effectively from the space. Our approach enables existing patch-based denoisers, which often find similar patches within an image window, to identify more similar patches by extending the limited search space into the entire space (i.e., all input images). We integrate our technique into state-of-the-art single-view denoising (block-matching and 3D filtering (BM3D)), and demonstrate that the BM3D combined with our approach is able to conduct multi-view image denoising effectively, without a major alteration to the existing algorithm.

INDEX TERMS BM3D, multi-view image denoising, patch embedding, similar patch selection.

I. INTRODUCTION

Image denoising is a well-established problem of estimating an unknown ground truth image from an observed noisy image corrupted by random artifacts (noise). A classical but common approach is neighborhood filtering (e.g., [22], [24]) that identifies similar pixels (e.g., the pixels with similar values) in a spatial neighborhood at each pixel and weight-averages the gathered similar pixels to reduce the noise in the input image. However, it is typically challenging to identify similar pixels correctly since the similarity between two pixels (e.g., the squared difference between two pixel colors) can also be noisy as the distance computation relies on noisy pixel values.

A more robust denoising approach against noise is to collect similar pixels using a patch-wise distance in a spatial neighborhood. For example, non-local means denoising [2] exploits an l_2 distance between two image patches, which can be more robust than a pixel-wise distance. Block-matching and 3D filtering (BM3D) [9] takes the same principle (patch-based computation), but the technique employs collaborative filtering that denoises identified similar patches simultaneously in a sparse 3D transform domain.

Common to these approaches is to identify similar pixels (or patches) in a spatial neighborhood, and the neighborhood pixels are typically chosen as the pixels in a local image window. The underlying assumption of this local window search is that statistically similar values (e.g., similar ground truth values) are usually located in a *spatially similar* image region. This assumption may work well for single image denoising, but it can be more desirable to find similar values across multiple images when multi-view images are given to a denoiser as input.

However, applying the local window search directly to the multi-view denoising scenario is not straightforward since directly extending a 2D spatial neighborhood into a 3D volume is not often effective because the images can be captured from different viewpoints. A common approach for multi-view image denoising is finding similar patches across images using additional information (e.g., depth or disparity) [13], [14], [18]. Estimating the depth or disparity can be a suitable choice, particularly if input images are captured from similar viewpoints (e.g., stereo cameras). Recently, deep-learning approaches demonstrated outstanding performance, given a set of images whose viewpoints are similar

The associate editor coordinating the review of this manuscript and approving it for publication was Ikramullah Lali.

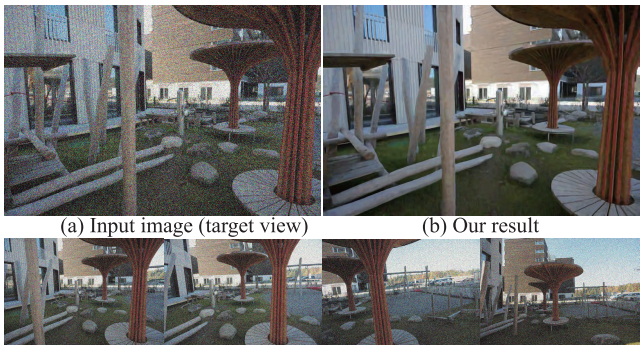


FIGURE 1. Our multi-view denoising result. We find similar patches both in the target view image (a) and in the other input images (c) taken from different viewpoints so that BM3D combined with our technique (b) denoises the target image more effectively. The input images ((a) and (c)) are corrupted by Gaussian noise with $\sigma = 50$.

to each other (e.g., a burst of images taken from a mobile device). However, if the views are very different, accurately estimating the information is highly difficult or not feasible. Hence, robustly handling more general scenarios, where input images are taken from arbitrarily different positions, remains a challenge.

This paper proposes a new approach for multi-view image denoising, which does not restrict the inputs (e.g., images taken from similar viewpoints). Our technique relies on neither prior information nor the heavy training process of neural networks, unlike the previous multi-view approaches. We instead directly adapt the state-of-the-art single-view denoising (BM3D) to be capable of exploiting multi-view images in its denoising process. Our high-level approach is to find similar patches in the entire input domain (i.e., all image patches from multi-view images) instead of a limited space (i.e., only a spatial neighborhood) so that BM3D can perform its collaborative filtering more effectively. To fulfill our high-level idea, we embed all image patches into a low-dimensional space and search for similar patches in a low-dimensional data structure (e.g., kd-trees). While our extension to the single-view denoising is simple and requires minimal modification (i.e., a search space) to the method, our method is able to improve the state-of-the-art given general multi-view scenarios where we cannot assume similar viewpoints (e.g., Fig. 1).

II. RELATED WORK

A. SINGLE-VIEW IMAGE DENOISING

Image denoising has been extensively studied in the image processing field, and its ultimate goal is to restore the (unknown) ground truth image from a single image corrupted by random artifacts (i.e., noise).

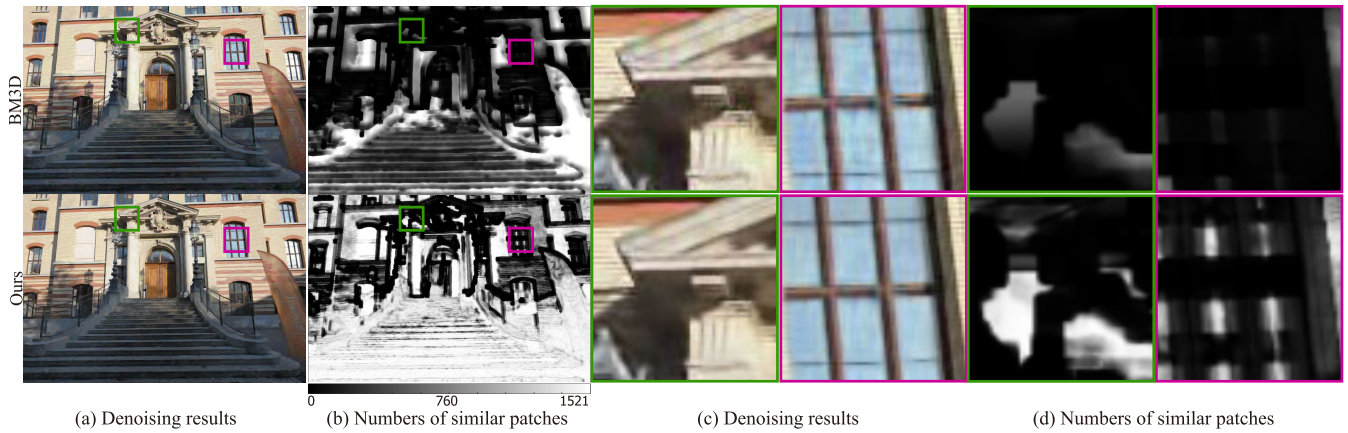
Well-known examples include anisotropic diffusion [23], bilateral filter [22], non-local means [2], wavelet thresholding [4], and regression-based approaches [21]. In addition to these conventional methods, deep learning-based approaches have been actively studied [3], [8], [11], [25], aiming to train a neural network that estimates the clean image from an observed noisy image. Although the recent deep learning models have shown impressive denoising performance, block-matching and 3D filtering (BM3D) [9] is still considered a state-of-the-art denoiser [7].

BM3D [9] uses a collaborative filtering that transforms similar patches together as a group and, thereby, effectively removes noise while preserving high-frequency details in each image patch. Due to the computational cost, BM3D explores only a neighborhood (called a search window) of each patch to find similar patches, and hence its denoising quality highly depends on whether there are a sufficient number of similar patches in the spatial neighborhood. If this is not the case, BM3D tends to retain residual noises with high variances. To remedy this issue, this paper suggests a novel method of finding similar patches in the entire domain (i.e., multiple input images), not just within a limited search window, which can improve the denoising quality of BM3D.

B. MULTI-VIEW IMAGE DENOISING

When multiple images are taken for the same objects yet with different views, it is desirable for an image denoiser to exploit all the input images together for better denoising quality. This problem is called multi-view image denoising. What is most challenging in this problem is how to collect similar pixels or patches across different images. One typical solution is to estimate additional information such as depth map or disparity map and match pixels relevant to each other (e.g., pixels with a similar depth) across images [13], [14], [18]. This approach can be ideal if we are already given the information or can calculate per-pixel depth accurately (e.g., in the case of stereo images). Otherwise, one should estimate the depth only from the noisy color images without any prior (e.g., similar views), and this estimation problem is as challenging as the denoising problem, particularly for the images with quite different views (e.g., Fig. 1).

Burst denoising can also be seen as a special case of multi-view image denoising, where we deal with a sequence of photos usually taken by mobile devices in a moment. With this sequence of images with almost the same views, burst denoising can outperform single image denoising in terms of denoising quality. For example, recent approaches [16], [17] exploited an end-to-end neural network that blends pixel colors across images without explicitly estimating the additional information (i.e., depth). These approaches dealt with misalignment in multiple images through a deep network. However, it still remains challenging to perform better than the state-of-the-art single-view denoising (BM3D) for a general scenario where the views are not similar to each other. Unlike the existing approaches, our method relies on neither depth estimation nor a neural network. We instead search for candidate patches across all images from an embedding space and feed retrieved candidates into BM3D so that the single-image denoiser can naturally be extended to multi-view image denoising.



(a) Denoising results (b) Numbers of similar patches (c) Denoising results (d) Numbers of similar patches

FIGURE 2. BM3D denoising results (a) without (top) and with our technique (bottom). We visualize the number of similar patches per pixel given the same user threshold (τ) for both approaches (b). We set the number of candidate patches for our approach to be the same as the one used in BM3D (e.g., $1521 = 39 \times 39$). BM3D finds similar patches whose patch-wise l_2 distance from the reference patch is less than τ in a spatial window (e.g., 39×39). When it fails to find enough numbers of similar patches, the denoising result leaves low-frequency artifacts (see the top row in (c) and (d)). Our technique increases the number of similar patches by feeding a set of candidate patches identified from the entire inputs (multi-view images) to BM3D, and it allows the method to produce an improved result (see the bottom row in (c) and (d)).

III. BACKGROUND

1) PROBLEM STATEMENT

Let us consider a commonly used noise model for image denoising:

$$y(x) = \mu(x) + \epsilon(x), \tag{1}$$

where $y(x)$ is an observed value (e.g., the pixel color) at pixel position x. $\mu(x)$ is the unknown ground truth color to be estimated and $\epsilon(x)$ is assumed to be i.i.d Gaussian noise with variance σ^2, i.e., $\epsilon(x) \sim N(0, \sigma^2)$. Image denoising is a problem of estimating the unknown $\mu(x)$ only from the observed noisy signal $y(x)$.

2) BLOCK-MATCHING AND 3D FILTERING (BM3D)

Given the noise model above, we present a brief explanation of how BM3D [9] works as our method is an extended version of BM3D for multi-view image denoising. BM3D performs two filtering steps at a high level, where each step equally consists of 1) grouping by a block-matching, 2) collaborative filtering, and 3) aggregation.

Grouping is a process of collecting similar patches for each patch. More formally, let P_c of size $n_P \times n_P$ denotes a reference patch being processed at pixel c (e.g., 8×8 pixels located top-left at pixel c). Then, for each P_c, BM3D finds the set S_c of similar patches whose distances (i.e., dissimilarity) from P_c are small enough. To this end, it first constructs the candidate set Ω_c consisting of all the patches within a local search window (e.g., 39×39 image window) centered at c. Among the candidates, BM3D selects the similar patches S_c whose distances are less than a user-defined threshold τ:

$$S_c = \left\{ P_x \in \Omega_c \mid d^2(P_c, P_x) \leqslant \tau \right\}, \tag{2}$$

where $d^2(P_c, P_x)$ is an l_2 distance that defines the dissimilarity between two patches:

$$d^2(P_c, P_x) = \frac{\|P_c - P_x\|_2^2}{(n_P)^2}. \tag{3}$$

Once we construct S_c for each patch P_c, BM3D conducts a 3D transformation to the stacked similar patches (i.e., 3D image patches) and performs a collaborative filtering (a hard-thresholding and Wiener filtering in the first and second steps respectively) for the transformed patches so that the grouped patches are filtered together. As the filtered image patches can be overlapped, we can have multiple estimates for a pixel. Therefore, the last process of BM3D is an aggregation that takes the average of the multiple values to determine the final pixel value.

3) MOTIVATION OF OUR WORK

BM3D denoises the noisy image signal $y(x)$ effectively while preserving high-frequency details of an input image, but its denoising can only be successful when a sufficient number of similar patches S_c (i.e., the candidate patches whose distances are smaller than a threshold τ in Eq. 2) can be found per each reference patch. Otherwise, its resulting images can be under-blurred and leave remaining noise, as shown in Fig. 2. Technically, it occurs when the candidate patches in Ω_c nearby a reference point c do not contain a necessary number of similar patches. One can simply increase the threshold τ so that a larger number of similar patches can be involved in the collaborative filtering process, but it typically introduces over-blurred results due to an increased bias caused by less similar patches. A better approach is to select similar patches in the entire input domain (e.g., all the image patches) instead of a limited area (a local search window centered at the reference point c) so that a large number of similar patches can be found without increasing the threshold. This approach

can be beneficial, especially when multi-view images are employed. For example, we can increase the possibility of finding similar patches, especially for reference patches (e.g., in Fig. 2) that are not similar to the patches nearby the references by searching all the input patches even in the other views. In the next section, we present our proposed method of finding similar patches in the entire domain to enhance the performance of BM3D by remedying the under-blurred areas caused by lack of patch similarity.

IV. EXTENDED BM3D FOR MULTI-VIEW IMAGE DENOISING

This section presents our technique that selects similar image patches (S_c in Eq. 2) with respect to a reference patch P_c from the multi-view input images (not just from a local search window in a target view), allowing BM3D [9] to perform an effective multi-view denoising. This scheme may be considered a straightforward extension to the state-of-the-art single-view denoiser, but it can be challenging to efficiently find similar patches over a complete set of possible patches in all the images.

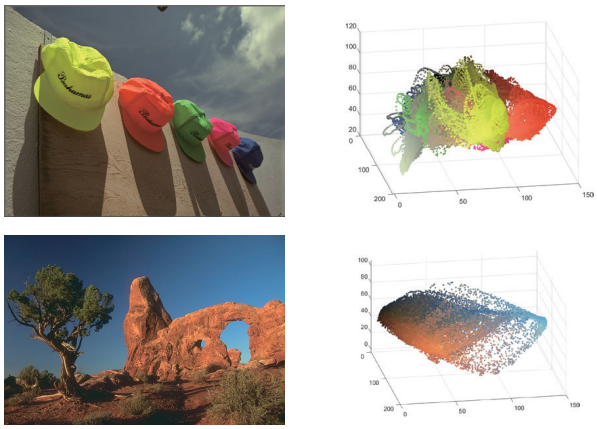
A naïve approach would be a brute-force search that first computes all the pairwise distances between patches across all input images and then take similar patches within the threshold in the order of their distances from each reference patch. It, however, is not a practical solution since the time complexity of the patch-wise distance, $d^2(P_c, P_x)$, is a square of the total number of pixels over all the images.

A potentially better approach is to build a hierarchical structure (e.g., a kd-tree) on all the patches and then to find candidate patches Ω_c for a reference patch using K-nearest neighbor (KNN) search ($K = |\Omega_c|$). Then, we can select the similar patches S_c, which are a subset of the candidates Ω_c (Eq. 2). Unfortunately, this approach suffers from the curse of dimensionality as image patches are in a high dimensional space (e.g., 64×3 for a colored 8×8 patch). Due to the high dimensionality of each patch, KNN search on the kd-tree over all the patches is neither efficient nor effective [1].

Our key strategy to address the challenges discussed above is to project each high-dimensional point (i.e., image patch) into a low-dimensional space while their pairwise similarities in the mapped space are relatively well preserved (Sec. IV-A). Then, we can efficiently perform KNN search with a kd-tree built upon this set of low-dimensional points (Sec. IV-B).

A. IMAGE PATCH EMBEDDING USING FastMap

We adopt one of the simplest embedding algorithms, FastMap [5], to embed all image patches into a k-dimensional space while maintaining the distance between two embedded patches to be close to the one between the patches in the original space. Specifically, we take all patches $P_1, \ldots, P_N$ (N is the total number of patches from all images) as input and then compute embedded patches $P^e_1, \ldots, P^e_N$. For example,



(a) (b)

FIGURE 3. Image patches in input images (a) are embedded into 3D space (b) using FastMap [5]. We set the color of the embedded points to the average color of input patches, and this result indicates that input patches with similar colors are embedded into a similar position in the low-dimensional space. The two input images are from Kodak24 [6] and BSD68 [20].

the i-th image patch, P_i is mapped into an embedded point $P^e_i = (P^e_{i,1}, \ldots P^e_{i,k})$ in k-dimensional space.

FastMap is an iterative algorithm that computes the j-th coordinate for all the patches at j-th iteration ($j = 1, \ldots, k$). Specifically, the j-th coordinate $P^e_{i,j}$ for the i-th patch is computed as

$$P^e_{i,j} = \frac{d^2_j(P_i, P_a) + d^2_j(P_a, P_b) - d^2_j(P_i, P_b)}{2d_j(P_a, P_b)}, \tag{4}$$

where $d^2_j(\cdot, \cdot)$ is the distance function between two patches at j-th iteration. The distance function $d^2_j(P_i, P_a)$ is updated recursively at each step j:

$$d^2_{j+1}(P_i, P_a) = d^2_j(P_i, P_a) - (P^e_{i,j} - P^e_{a,j})^2. \tag{5}$$

For the recursion above, the distance function at the first step, $d^2_1(\cdot, \cdot)$, is set to the l_2 patch-wise distance used in BM3D (Eq. 3). Note that the distance function used for BM3D is a non-negative and symmetric function satisfying the triangle inequality, and thus it satisfies the conditions required in FastMap [5].

The two pivot patches, P_a and P_b, are chosen such that their distance $d^2_j(P_a, P_b)$ is maximized. Exactly computing the pivot patches P_a and P_b in Eq. 4 is computationally expensive ($\mathcal{O}(N^2)$), and thus the algorithm uses a simple heuristic. For example, it selects a patch randomly and finds P_b that has the largest distance from the randomly chosen one. Then, the method picks the patch P_a whose distance is the largest from P_b. We repeat this process five times and select the two pivots with the largest distance, like the original method [5].

The overall complexity of FastMap is $\mathcal{O}(kN)$ and thus we can embed all patches into k-dimensional space efficiently. As shown in Fig. 3, image patches with similar colors are located closely in the embedding space even if we use a low dimensionality (i.e., $k = 3$), since FastMap conducts a

dimensionality reduction while preserving the original distance between patches. We apply the embedding technique to the main denoising stage in BM3D. Specifically, we take the images pre-filtered by BM3D (i.e., the output by the first step of BM3D) as input, and compute embedded points of all input patches so that its main denoising can be performed with refined similar patches identified in the embedding space (Sec. IV-B).

B. SIMILAR PATCH SELECTION IN EMBEDDING SPACE

Once we compute embedded patches from multiple input images, we build a kd-tree using the coordinates of the patches. In the main denoising stage of BM3D, we search $K = |\Omega_c|$ embedded points close to the embedded point of the reference patch P_c in the kd-tree through K-nearest neighbor (KNN) search. Let us denote the candidate patches collected using the KNN search by Ω_c^e. Our adaptation to BM3D is to replace the local patch candidates Ω_c (in Eq. 2), which are located in a local search window, with our global patch candidates Ω_c^e:

$$S_c = \left\{ P_x \in \Omega_c^e \mid d^2(P_c, P_x) \leqslant \tau \right\}. \tag{6}$$

While our modification to the original BM3D is simple, it allows the single-view denoiser to exploit potentially much more similar patches, even in the other-view images. For example, the denoising quality of BM3D can be degraded when the number of similar patches is not enough within a local search window in a target-view image that we want to denoise. Our integration to BM3D is able to enhance the denoising quality of BM3D for this scenario, as we can increase the possibility of finding similar patches by exploiting the images captured by different camera viewpoints, as shown in Fig. 2.

V. RESULTS AND DISCUSSION

This section compares our method with the baseline (BM3D) that denoises each image independently. We have implemented our extension on top of a C/C++ implementation of BM3D [12]. Also, we have set the parameters (e.g., patch sizes) for BM3D, as recommended by the original paper [9]. Our technique is also compared with a video extension of BM3D, VBM4D [15], and recent burst denoising using kernel predicting network (KPN) [17]. For a fair comparison with the learning-based approach (KPN), we have retrained it with a batch size of 64 for 250K iterations using the Open Image dataset [10] while varying the noise level in images from 5 to 50.

We use five images for each test scene and report quantitative results using the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [26]. Specifically, we simulate additive white Gaussian noise (AWGN) by varying its variance (i.e., noise level). All experiments have been conducted on a PC with an AMD Ryzen Threadripper 3990X CPU (2.9 GHz) for all tests.

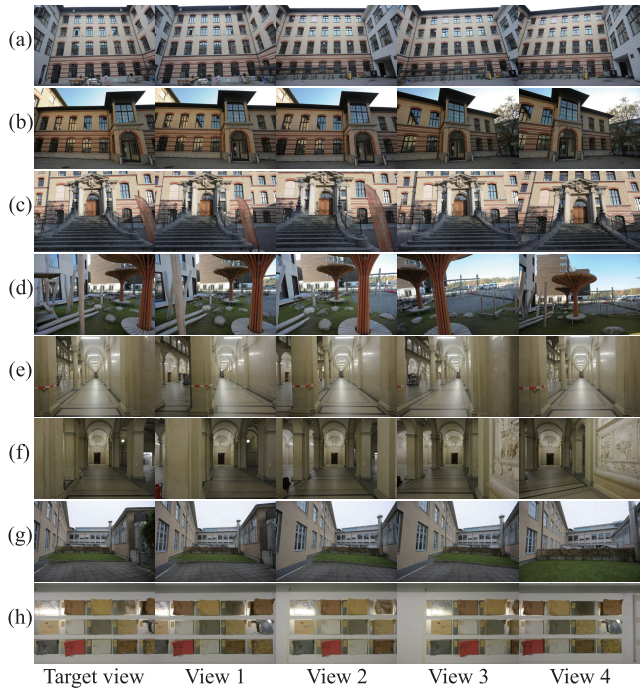


FIGURE 4. Test scenes from ETH3D [19] dataset. (a): Courtyard, (b): Facade 1, (c): Facade 2, (d): Playground, (e): Relief 1, (f): Relief 2, (g): Terrace, and (h): Train.

1) DATASET

We use the ETH3D [19] dataset for multi-view denoising experiments, and the dataset consists of multiple images captured from various camera positions and angles. Note that the dataset is not dedicated to multi-view scenarios, and thus we use only the eight scenes that can be categorized as multi-view denoising scenarios. Specifically, each scene contains five images, and we treat one image per scene as a target image that should be denoised. We use 810×540 for the image resolution. Fig. 4 shows the eight scenes, each of which has five images with different views. As shown in the figure, views (positions and directions) are not very similar to each other, and thus identifying similar patches for this scenario can be challenging.

2) COMPARISONS UNDER HOMOGENEOUS NOISE CONDITIONS

Table 1 shows the numerical accuracy of the tested methods for the eight scenes (in Fig. 4), where we use homogenous noise levels. Specifically, we test the three levels of noise, $\sigma = 30$, 50, and 70, and the simulated noise is added in the muti-view images (not only the target but also the other images). Also, we compare the techniques qualitatively in Fig. 5.

Existing multi-view denoisers (VBM3D and KPN) generate over-blurred artifacts or leave some low-frequency noise. Those can be very effective when the input images have very similar viewpoints, but the tested scenarios are far from the case. The baseline method (BM3D) outputs more visually

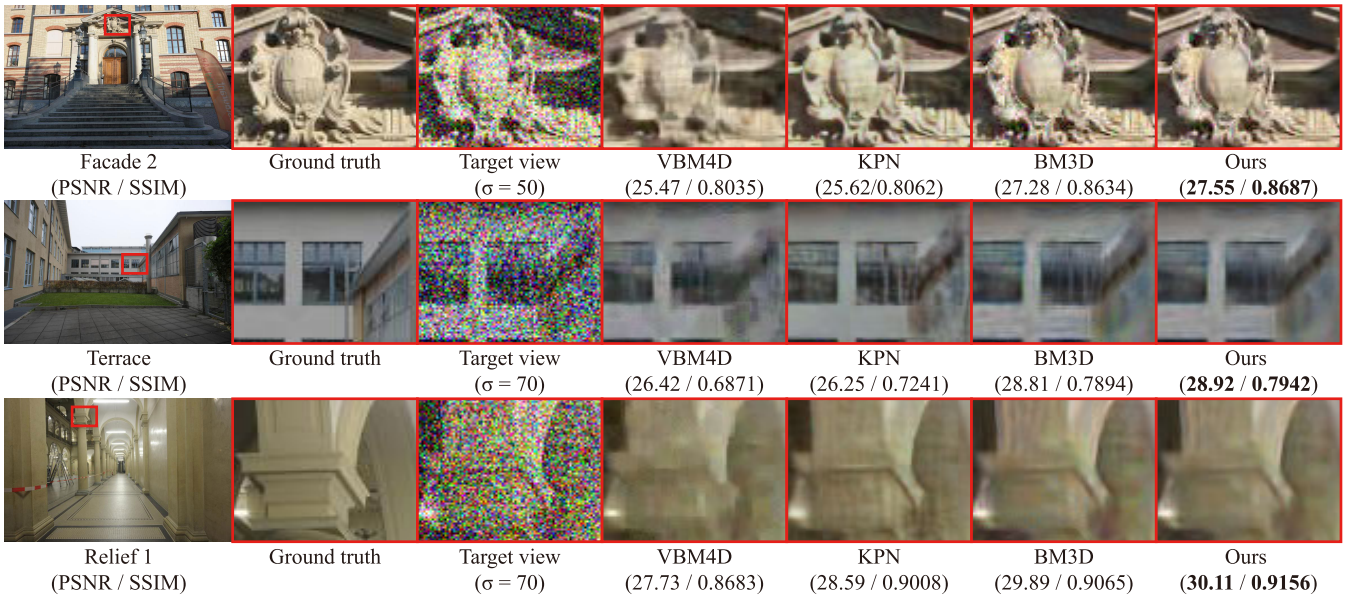


FIGURE 5. Multi-view denoising results when the same amount of noise is added to the multiple input images (five views shown in Fig. 4). The tested methods denoise a target-view image (in the third column) where we vary the noise level, σ.

TABLE 1. Quantitative results of the tested denoisers. We vary the noise level σ in the input multi-view images from 30 to 70 and measure the PSNR and SSIM values of the denoised target-views from each tested method. The last column shows the averages of the numerical errors for the eight scenes.

Scenes	Courtyard	Facade 1	Facade 2	Playground	Relief 1	Relief 2	Terrace	Train	Average
Noise level	$\sigma = 30$								
VBM4D	27.87 / 0.8245	29.04 / 0.8727	27.53 / 0.8722	29.16 / 0.8597	31.56 / 0.9335	32.64 / 0.9312	30.15 / 0.8301	31.45 / 0.9368	29.93 / 0.8826
KPN	27.89 / 0.8282	28.78 / 0.8752	27.70 / 08754	29.09 / 0.8513	32.29 / 0.9466	32.97 / 0.9333	30.53 / 0.8381	31.42 / 0.9381	30.08 / 0.8858
BM3D	30.22 / **0.8976**	31.40 / 0.9158	29.88 / 0.9162	**31.65 / 0.9121**	**34.07** / 0.9589	**35.38** / 0.9607	**32.75 / 0.8945**	**34.31 / 0.9595**	32.46 / **0.9269**
Ours	**30.34** / 0.8973	**31.50 / 0.9164**	**30.00 / 0.9164**	**31.65** / 0.9106	34.05 / **0.9595**	35.35 / **0.9611**	32.71 / 0.8919	34.21 / 0.9589	**32.48** / 0.9265
Noise level	$\sigma = 50$								
VBM4D	25.71 / 0.7341	26.85 / 0.8150	25.47 / 0.8035	26.94 / 0.7869	29.31 / 0.8993	30.15 / 0.8903	27.93 / 0.7516	28.97 / 0.8949	27.67 / 0.8220
KPN	26.25 / 0.7474	26.87 / 0.8240	25.62 / 0.8062	27.05 / 0.7854	30.19 / 0.9229	31.24 / 0.9124	28.43 / 0.7734	29.42 / 0.9145	28.13 / 0.8358
BM3D	27.83 / 0.8348	29.19 / 0.8764	27.28 / 0.8634	29.24 / 0.8654	31.52 / 0.9322	32.83 / 0.9346	30.37 / 0.8360	31.59 / 0.9234	29.98 / 0.8833
Ours	**28.12 / 0.8372**	**29.51 / 0.8825**	**27.55 / 0.8687**	**29.44 / 0.8687**	**31.74 / 0.9374**	**33.11 / 0.9403**	**30.52 / 0.8400**	**31.74 / 0.9310**	**30.21 / 0.8882**
Noise level	$\sigma = 70$								
VBM4D	24.18 / 0.6583	25.31 / 0.7594	24.04 / 0.7396	25.44 / 0.7221	27.73 / 0.8683	28.53 / 0.8558	26.42 / 0.6871	27.26 / 0.8508	26.11 / 0.7677
KPN	25.09 / 0.6909	25.50 / 0.7865	24.13 / 0.7454	25.57 / 0.7285	28.59 / 0.9008	29.21 / 0.8764	26.25 / 0.7241	27.69 / 0.8870	26.50 / 0.7925
BM3D	26.35 / 0.7785	27.70 / 0.8395	25.76 / 0.8162	27.68 / 0.8224	29.89 / 0.9065	30.95 / 0.9051	28.81 / 0.7894	29.98 / 0.8899	28.39 / 0.8434
Ours	**26.66 / 0.7831**	**27.94 / 0.8466**	**26.00 / 0.8240**	**27.85 / 0.8267**	**30.11 / 0.9156**	**31.26 /0.9157**	**28.92 / 0.7942**	**30.05 / 0.9023**	**28.60 / 0.8510**

pleasing and accurate results than the VBM3D and KPN. On the other hand, our method enables BM3D to boost its denoising quality for most cases both numerically and visually. For example, our method improves the denoising quality of BM3D up to 0.32 dB for the Facade 1 scene in Table 1. Technically, our technique improves the BM3D, especially for the problematic regions where similar patches cannot be found within a spatial neighborhood (see Fig. 5).

3) COMPARISONS UNDER HETEROGENEOUS NOISE CONDITIONS

Table 2 and Fig. 6 also show the results of the denoising methods for the multi-view denoising scenarios, but we relax the homogenous noise condition. Precisely, we set the noise levels for the other views to 10 (for the first), 20 (second), 30 (third), and 40 (last). We then only vary the noise level in the target view from 30 to 70.

Analogously in the test with homogeneous noise conditions, it is observed that our method improves the baseline method (BM3D) for most cases, but our improvement can become more significant when the noise in the other views is smaller than that in the target view (e.g., $\sigma = 70$ for the target view). For example, our numerical improvement over BM3D is up to 0.70 dB for the Relief 2 scene in Table 2. Technically, our method enables BM3D to exploit less noisy patches from the other views, especially when the target view is severely noisy.

We have also observed that our numerical improvement for BM3D can become more significant as the number of views increases. For example, our PSNR improvements are 0.17 dB (only with the target view), 0.37 dB (with the target and first views), and 0.45 dB (with all the views) on average, given the tested scenes with $\sigma = 70$.

4) ANALYSIS ON THE DIMENSIONALITY OF OUR EMBEDDING SPACE

In Fig. 7, we measure the numerical accuracy of our method by varying the dimensionality of our embedding space.

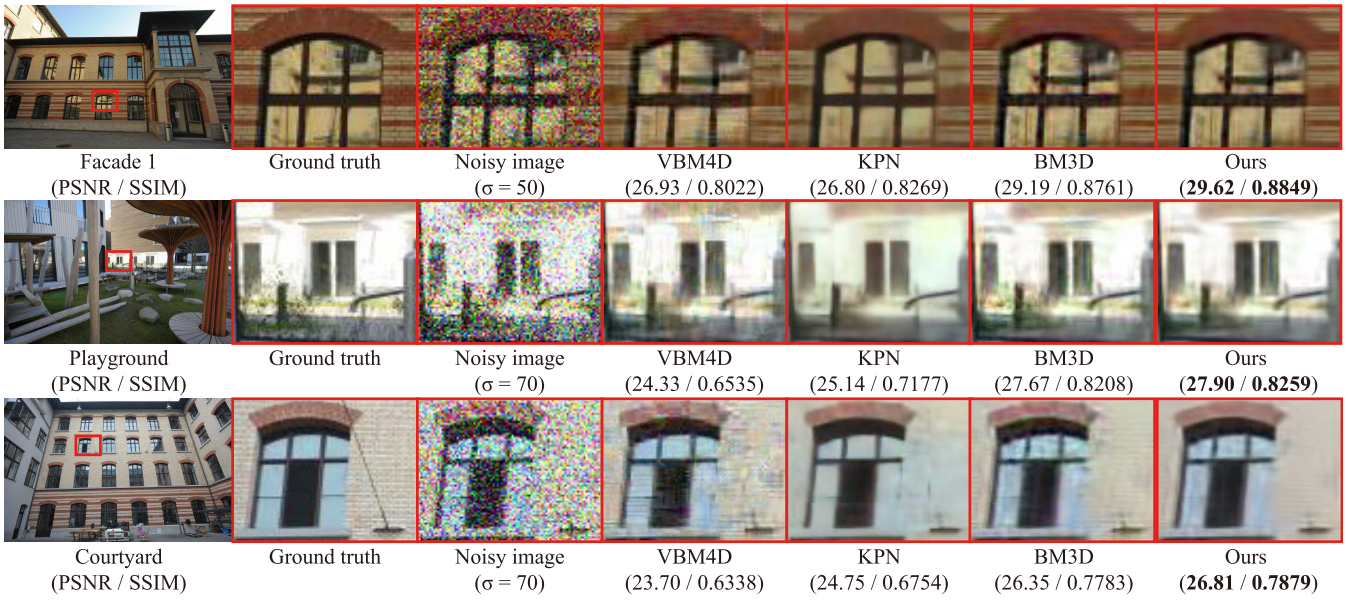


FIGURE 6. Multi-view images denoising results on the heterogeneous noise conditions where the noise in the target-views is different from the other views. We set the noise levels for the other views to $\sigma = 10$ (for the first view), 20 (second), 30 (third), and 40 (the last view in Fig. 4), respectively.

TABLE 2. Quantitative results of the tested denoising methods for the eight scenes, where use different amounts of noise for each view.

Scenes	Courtyard	Facade 1	Facade 2	Playground	Relief 1	Relief 2	Terrace	Train	Average
Noise Level	$\sigma = 30$								
VBM4D	28.14 / 0.8315	29.28 / 0.8742	27.73 / 0.8755	29.30 / 0.8615	31.64 / 0.9327	32.71 / 0.9293	30.39 / 0.8302	31.62 / 0.9323	30.10 / 0.8834
KPN	27.97 / 0.8283	28.96 / 0.8801	27.75 / 0.8770	29.21 / 0.8552	32.37 / 0.9490	33.49 / 0.9425	30.81 / 0.8484	31.54 / 0.9412	30.26 / 0.8902
BM3D	30.20 / **0.8964**	31.38 / 0.9149	29.85 / 0.9163	31.65 / **0.9122**	34.08 / 0.9590	35.44 / 0.9607	32.69 / **0.8933**	**34.29** / 0.9593	32.45 / 0.9265
Ours	**30.33** / 0.8951	**31.53 / 0.9158**	**30.01 / 0.9171**	**31.70** / 0.9112	**34.16 / 0.9611**	**35.59 / 0.9625**	**32.74** / 0.8927	34.23 / **0.9606**	**32.53 / 0.9270**
Noise Level	$\sigma = 50$								
VBM4D	25.73 / 0.7307	26.93 / 0.8022	25.70 / 0.8039	26.80 / 0.7737	28.67 / 0.8720	29.32 / 0.8534	27.74 / 0.7074	28.60 / 0.8429	27.44 / 0.7983
KPN	26.05 / 0.7255	26.80 / 0.8269	25.58 / 0.7957	26.76 / 0.7736	29.82 / 0.9257	31.23 / 0.9214	28.38 / 0.7755	28.84 / 0.9120	27.93 / 0.8320
BM3D	27.87 / 0.8356	29.19 / 0.8761	27.28 / 0.8640	29.18 / 0.8640	31.51 / 0.9316	32.81 / 0.9330	30.35 / 0.8371	31.64 / 0.9244	29.98 / 0.8832
Ours	**28.28 / 0.8395**	**29.62 / 0.8849**	**27.70 / 0.8732**	**29.41 / 0.8673**	**32.01 / 0.9422**	**33.45 / 0.9449**	**30.63 / 0.8457**	**31.98 / 0.9405**	**30.38 / 0.8923**
Noise Level	$\sigma = 70$								
VBM4D	23.70 / 0.6338	24.54 / 0.6981	23.52 / 0.6979	24.33 / 0.6535	25.56 / 0.7625	25.94 / 0.7291	25.03 / 0.5539	25.46 / 0.6996	24.76 / 0.6786
KPN	24.75 / 0.6754	25.17 / 0.7763	24.06 / 0.7371	25.14 / 0.7177	27.86 / 0.9052	29.32 / 0.8965	26.76 / 0.7317	26.90 / 0.8819	26.25 / 0.7902
BM3D	26.35 / 0.7783	27.69 / 0.8410	25.73 / 0.8147	27.67 / 0.8208	29.86 / 0.9046	31.00 / 0.9062	28.86 / 0.7896	29.99 / 0.8902	28.39 / 0.8432
Ours	**26.81 / 0.7879**	**28.10 / 0.8533**	**26.07 / 0.8280**	**27.90 / 0.8259**	**30.52 / 0.9243**	**31.70 / 0.9263**	**29.17 / 0.8063**	**30.42 / 0.9187**	**28.84 / 0.8588**

As shown in the figure, there is a noticeable increase in PSNR values when changing the dimensionality from one to three. It indicates that too low dimensionalities (e.g., one) lead to a sub-optimal selection of similar patches, but the increase in the numerical accuracy from three to five is not drastic. Also, the computational time of our method increases from 64.68 s to 76.81 s when changing the dimensionality from three to five. As a compromise between the computational overhead and denoising quality, we have chosen a relatively low dimensionality (three).

5) COMPUTATIONAL OVERHEAD OF BM3D WITH OUR METHOD

Table 3 shows the runtime overheads of BM3D with and without our method. Precisely, we measure the computational times of both approaches when a target image is denoised. Unlike the original BM3D, the BM3D with ours takes five multi-view images of 810×540 size as input. As we construct an embedding space using the pre-filtered results by the method (the first step of BM3D), applying the pre-filtering to all the input images is needed. This pre-filtering is the most expensive process of the BM3D with ours. Overall, our adaptation to the original BM3D increases its computational time from 11.92 s to 64.68 s.

TABLE 3. Computational overheads of BM3D with and without our technique, given input images of size 810 × 540.

Task	BM3D	BM3D w/ ours
Pre-filtering (the first step of BM3D)	5.77 s	29.41 s
Calculating an embedding space	-	15.97 s
Main denoising (the second step of BM3D)	6.15 s	19.30 s
Total	11.92 s	64.68 s

Nevertheless, the increasing rate in the computational overhead caused by our adaptation can be decreased for a typical multi-view denoising scenario where all the input images (not just a target image) are denoised since we do not need to compute a separate embedding space for each target image.

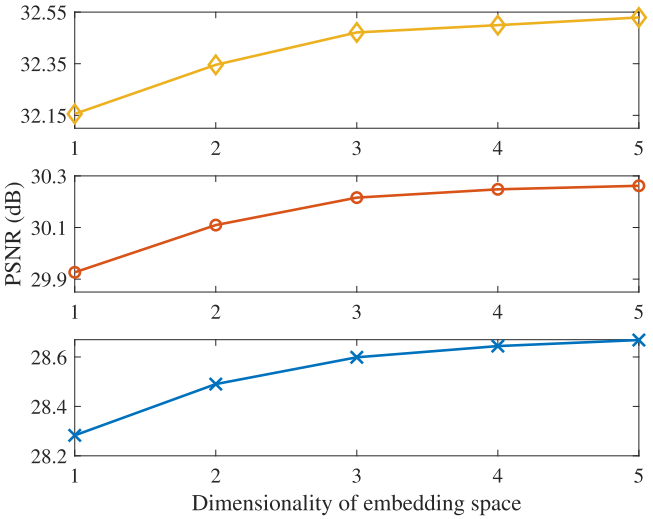


FIGURE 7. Numerical errors of the BM3D with our method, where we vary the dimensionality of embedding space. The PSNR values are the averages for the eight scenes where we use the homogeneous noise levels ($\sigma = 30$ in the top, 50 in the second, and 70 in the last row) for multi-view images (in Table 1).

For example, the original BM3D takes 59.65 s for denoising all the input images (i.e., five images), and BM3D with our adaptation requires 98.36 s. As a result, our computational overhead is less than $2\times$, compared to the time of the original BM3D.

6) LIMITATIONS AND FUTURE WORK

The main limitation of the proposed method is that our improvement for the original BM3D can be incremental when enough numbers of patches can be found in a spatial window. It technically restricts the overall improvement (e.g., PSNR values), as our method is only helpful when failing to find enough numbers of similar patches in a spatial neighborhood. Nonetheless, our adaptation to the method is simple, and our improvement can be significant locally in the problematic regions.

We would like to integrate our high-level idea of finding similar patches into other patch-based denoisers (e.g., non-local means [2]) as future work. Also, we want to extend our framework into a more general one that supports various noise types (e.g., correlated noise). In addition, more robust identification of similar patches using neural networks can be investigated.

VI. CONCLUSION

We have presented a simple extension to the state-of-the-art denoiser, BM3D, to improve its denoising quality for multi-view denoising by finding enough numbers of similar patches across multi-view images. We employ a well-known embedding technique (i.e., FastMap) that reduces the dimensionality of patches into a computationally tractable one (e.g., three) while taking the original patch distance of BM3D into account. It allows us to find similar patches effectively using a hierarchical structure (e.g., kd-trees) that maintains low-dimensional points (i.e., embedded patches). Our idea is simple and does not alter the main algorithm of the BM3D (including its parameters). We have demonstrated that our simple modification can improve the denoising quality of BM3D, especially in difficult image regions where similar patches cannot be collected within a small search window.

REFERENCES

[1] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ, USA: Princeton Univ. Press, 1961.
[2] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2005, pp. 60–65.
[3] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2392–2399.
[4] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 613–627, May 1995.
[5] C. Faloutsos and K.-I. Lin, "FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1995, pp. 163–174.
[6] R. Franzen. *Kodak Lossless True Color Image Suite*. Accessed: Oct. 16, 2020. [Online]. Available: http://r0k.us/graphics/kodak/
[7] B. Goyal, A. Dogra, S. Agrawal, B. S. Sohi, and A. Sharma, "Image denoising review: From classical to state-of-the-art approaches," *Inf. Fusion*, vol. 55, pp. 220–244, Mar. 2020.
[8] X. Jia, S. Liu, X. Feng, and L. Zhang, "FOCNet: A fractional optimal control network for image denoising," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6047–6056.
[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.
[10] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy. (2017). *OpenImages: A Public Dataset for Large-Scale Multi-Label and Multi-Class Image Classification*. [Online]. Available: https://github.com/openimages
[11] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2Void—Learning denoising from single noisy images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2124–2132.
[12] M. Lebrun, "An analysis and implementation of the BM3D image denoising method," *Image Process. On Line*, vol. 2, pp. 175–213, Aug. 2012.
[13] L. Zhang, S. Vaddadi, H. Jin, and S. K. Nayar, "Multiple view image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1542–1549.
[14] E. Luo, S. H. Chan, S. Pan, and T. Q. Nguyen, "Adaptive non-local means for multiview image denoising: Searching for the right patches via a statistical approach," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 543–547.
[15] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, "Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 3952–3966, Sep. 2012.
[16] T. Marinc, V. Srinivasan, S. Gül, C. Hellge, and W. Samek, "Multi-kernel prediction networks for denoising of burst images," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 2404–2408.
[17] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll, "Burst denoising with kernel prediction networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2502–2510.
[18] M. Miyata, K. Kodama, and T. Hamamoto, "Fast multiple-view denoising based on image reconstruction by plane sweeping," in *Proc. IEEE Vis. Commun. Image Process. Conf.*, Dec. 2014, pp. 462–465.
[19] T. Schops, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2538–2547.

[20] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vis.*, vol. 82, no. 2, pp. 205–229, Apr. 2009.
[21] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 349–366, Feb. 2007.
[22] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, 1998, pp. 839–846.
[23] J. Weickert, *Anisotropic Diffusion in Image Processing*. Stuttgart, Germany: Teubner, 1998.
[24] L. P. Yaroslavsky, *Digital Picture Processing: An Introduction*. Berlin, Germany: Springer, 1985.
[25] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.
[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

GEUNWOO OH (Student Member, IEEE) received the B.S. degree from the School of Robotics, Kwangwoon University, Seoul, South Korea, in 2016. He is currently pursuing the integrated M.S. and Ph.D. degrees with the School of Integrated Technology, Gwangju Institute of Science and Technology, Gwangju, South Korea. His research interests include denoising and rendering.

DONG-WAN CHOI (Member, IEEE) received the B.S. degree from Inha University, South Korea, in 2008, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2010 and 2014, respectively. From 2016 to 2017, he worked as a Research Associate with the Department of Computing, Imperial College London, U.K., and a Postdoctoral Fellow with the School of Computing Science, Simon Fraser University, Canada, from 2015 to 2016. He is currently an Assistant Professor with the Department of Computer Engineering, Inha University. He is also working on designing more efficient and lightweight deep learning models, and algorithms in a database perspective. His research interests include the areas of data mining, databases, big data algorithms, and scalable machine learning.

BOCHANG MOON (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), in 2010 and 2014, respectively. He was a Postdoctoral Researcher at Disney Research. He is currently an Assistant Professor with the Gwangju Institute of Science and Technology (GIST). His research interests include rendering, denoising, and augmented and virtual reality. He is a member of ACM. He served as a PC Member for international conferences, such as EGSR, I3D, PG, and CGI.