Towards learning-based denoising of light fields

Tomás Soares De Carvalho Feith, Michela Testolina, and Touradj Ebrahimi

Multimedia Signal Processing Group (MMSPG)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
tomas.soaresdecarvalhofeith@epfl.ch, michela.testolina@epfl.ch, touradj.ebrahimi@epfl.ch

ABSTRACT

In recent years, new emerging immersive imaging modalities, e.g. light fields, have been receiving growing attention, becoming increasingly widespread over the years. Light fields are often captured through multi-camera arrays or plenoptic cameras, with the goal of measuring the light coming from every direction at every point in space. Light field cameras are often sensitive to noise, making light field denoising a crucial pre- and post-processing step. A number of conventional methods for light field denoising have been proposed in the state of the art, making use of the redundant information coming from the different views to remove the noise. While learning-based denoising has demonstrated good performance in the context of image denoising, only preliminary works have studied the benefit of using neural networks to denoise light fields. In this paper, a learning-based light field denoising technique based on a convolutional neural network is investigated by extending a state-of-the-art image denoising method, and taking advantage of the redundant information generated by different views of the same scene. The performance of the proposed approach is compared in terms of accuracy and scalability to state-of-the-art methods for image and light field denoising, both conventional and learning-based. Moreover, the robustness of the proposed method to different types of noise and their strengths is reviewed. To facilitate further research on this topic, the code is made publicly available at https://github.com/mmspg/Light-Field-Denoising

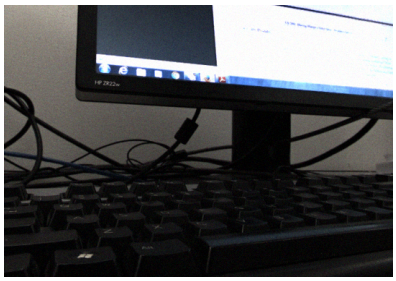
Keywords: denoising, light fields, immersive imaging, machine learning, deep learning

1. INTRODUCTION

Immersive imaging modalities are transforming the way visual information is captured, processed, and interacted with. As an example, virtual and augmented reality applications are assisting students during their education, and new techniques in the field of medicine, engineering, and psychology are being developed using this technology, providing realistic simulations of complex scenarios and environments. Moreover, in entertainment, immersive imaging modalities are enabling new forms of storytelling and interactive experiences. In this context, light field imaging is receiving increasing attention. Contrary to traditional cameras which use a 2D sensor capable of capturing only the intensity and color, light field cameras are also able to capture the direction and position of the light rays. This is often implemented through multi-camera arrays or, in the case of plenoptic cameras such as Lytro and Raytrix, by embedding micro-lenses arrays. This allows for capturing multiple views of the same scene at the same time, allowing, among others, manipulation of the perspective, focus, and depth of field of the image after its capture.

Similarly to traditional imaging technologies, light field photography is susceptible to noise. Notably, light fields are even more sensitive to camera noise as they rely on a more complex capture technology. In order to reduce the impact of this artifact, pre- and post-processing techniques, such as denoising, may be applied to images. While the state of the art in image denoising is fairly established, only a limited number of works have explored light field denoising. Moreover, while learning-based image denoising algorithms have demonstrated improved performance when compared to traditional methods, no effort has yet been devoted to the field of learning-based light field denoising.

In this paper, a novel method for light field denoising, developed on the basis of an existing learning-based image denoising method is proposed, and its performance is assessed by comparing it to existing state-of-the-art methods for conventional light-field denoising. Moreover, as image denoising algorithms can be trivially applied



(a) `backlight`

(b) `flower`

(c) `pens`

(d) `fence`

(e) `t-rex`

(f) `reflection`

Figure 1: SAIs from each of the selected testing scenes.

to light fields by processing each view separately, conventional and learning-based image denoising methods are additionally considered during the assessment of the performance.

The contributions of this paper are as follows:

- A light field dataset is presented, by merging multiple existing datasets. The size of the proposed dataset is suitable for training a deep-learning model, and a division between training, validation, and testing datasets is proposed. Moreover, a strategy for applying synthetic-realistic noise is presented.
- An optimized stitch-patching algorithm is presented, able to effectively search for the most similar patches among different views while being computationally efficient.
- A novel method for light-field denoising inspired by a state-of-the-art image denoising architecture is proposed.
- A comprehensive assessment of the performance of state-of-the-art methods for both image and light field compression, both conventional and learning-based is performed.

While this paper reports only preliminary results on the topic, the potential of these applications is emphasized, with the goal of inspiring further research on learning-based approaches to light field denoising.

2. RELATED WORK

2.1 Image denoising

Image denoising is a key pre- and post-processing step in many applications. A typical approach to image denoising is to consider noisy images as the sum of a noise map and an original noise-free image, where the main goal is to estimate the noise map and restore the original image.

Over the years, several classical methods have been proposed, e.g. methods based on wavelet thresholding, non-local self-similarity models,[1,2] sparse models,[3,4] or gradient models.[5,6] As an example, a method based on

block-matching and 3D filtering was introduced by Dabov *et al.* in 2007,[7] presenting a non-local self-similarity model used for image denoising, and often considered as a baseline for assessing the performance of the most recently-proposed methods. However, most of these classical methods suffer from a number of drawbacks: firstly, they tend to be time-consuming and computationally expensive;[8] secondly, they often rely on several hard-coded parameters, which makes it harder to achieve optimal performance and generalize over different types of noise.

Recently, with the success of machine learning techniques, several learning-based approaches have been proposed in the state of the art. Early work focused on discriminative models that attempted to learn image priors, e.g. Cascade of Shrinkage Fields[9] or Trainable Nonlinear Reaction Diffusion.[10,11] More recent work mainly focused on Convolutional Neural Networks (CNNs)[12–14] or Generative Adversarial Networks (GANs).[15,16] Notably, the usage of CNNs stemmed from its ability to exploit local information efficiently, while GANs surfaced as an alternative for handling real noisy images, for which deep CNNs can be insufficient.[17] As an example, the DnCNN method proposed by Zhang *et al.* in 2017,[18] presents a CNN-based model for image denoising, which aims at estimating the noise map from the noisy data, and successively subtracting it from the noisy image to get the clean image.

2.2 Light fields denoising

Contrary to the traditional image denoising, light Field denoising has been developed only recently. While all image denoising methods can be trivially applied to light field denoising by looking at each single Sub-Aperture Image (SAI), the ultimate goal of light field denoising is to use the redundant information present in multiple views to improve the quality of the captured light fields.

Previous classical attempts at light field denoising aimed at splitting and processing 2D light field slices,[19] or taking into account the 4D structure of light fields by using Gaussian Mixture Model light field patch priors.[20] An interesting approach consists in stacking the SAIs and using video denoising, where each SAI is considered to be one frame of the generated video.[21] In 2012 Alain and Smolic presented LFBM5D,[22] i.e. a generalization of the BM3D method, translated to the light field domain by searching for similar patches not only within the same image but also among neighboring views

Learning-based methods for light field denoising are scarce. One of the main methods available[23] uses anisotropic parallax analysis to guide the denoising process, which is performed via two CNNs trained jointly. However, there don't seem to be other alternative approaches, which suggests that this is a challenging problem, with potentially many directions for further explorations.

As light field denoising falls under the broader category of light field restoration and enhancement, there are closely related fields that should also be considered, and ideas from them can be translated into the denoising task. As an example, Fan *et al.* proposed in 2017 a method for light field superresolution,[24] which uses a 2-stage CNN. The strategy used to exploit the 4D structure of light fields, i.e. by applying a patch match algorithm to produce stitch-patched versions of an SAI from other views of the scene, which can be implemented in the context of light field denoising.

3. LIGHT FIELDS DATASET

In order to train the proposed learning-based light field denoising model and to assess its performance, a large dataset of SAIs was constructed by merging the data from a number of datasets already available in the state of the art. The utilized datasets, as well as any post-processing operation applied to its images, are described in Table 1.

Light fields are normally stored using the ESLF and LFR formats. Notably, LFR files contain the raw lenslet image from the camera, and ESLF files decode the light field into an orthogonal grid of lenslet images. To extract the data contained in the ELSF file into a more common *png* file, we used the MATLAB Light Field Toolbox,[25] while extraction from LFR format was done using the Plenopticam package.[26]

The full dataset has a total of 409 scenes, with 23'301 images, and includes natural scenes, geometrical shapes, synthetic images, and various lighting conditions. The dataset was divided into a training set with 400 scenes, a validation set including 32 scenes, and a testing set representing 6 scenes, as depicted in Figure 1. During training, synthetic Gaussian noise of random strength was added to each scene.

Table 1: Description of datasets used.

Dataset	Scenes	Post-Processing Applied
Stanford Lytro Light Field Archive[27]	212	Decoding ESLF format to SAI grid
INRIA Synthetic Light Fields Dataset[28]	91	None
EPFL Light Field Dataset[29]	60	Decoding LFR format to SAI grid
4D Light Field Benchmark[30,31]	28	None
Stanford Light Field Archive[32]	10	None
MIT Synthetic Light Fields Dataset[33–36]	8	None

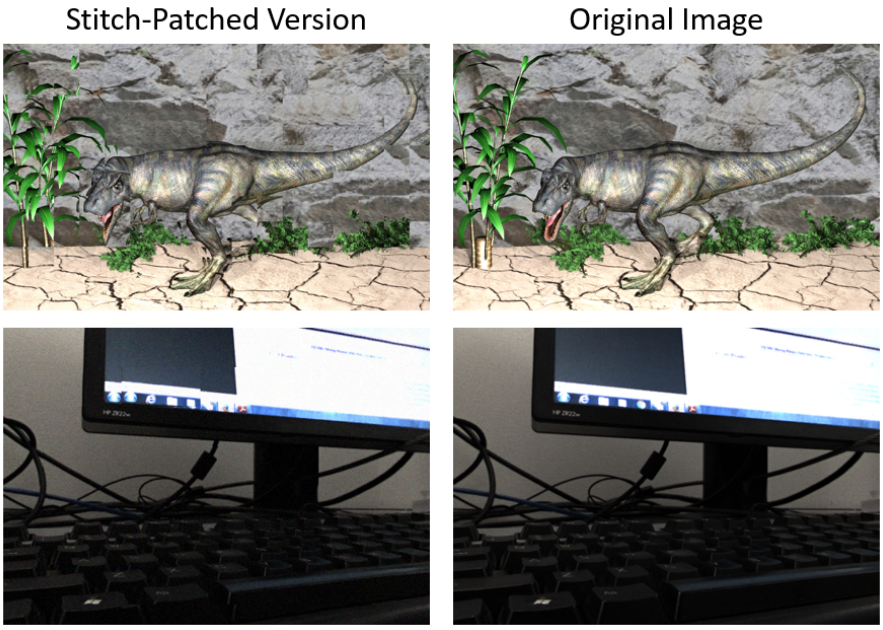


Figure 2: Examples of the original image and stitch-patched version for two scenes. The top scene has wide angular spacing (so the local similarity isn't enough to build a coherent image) and the bottom has a short angular spacing (locally similar patches fit well together to form a globally similar image).

4. LEARNING-BASED LIGHT FIELDS DENOISING

The learning-based light field denoising method proposed in this paper uses the DnCNN model[18] as a baseline, presenting a possible extension for light fields that exploits the redundant information present in neighboring images. The proposed method is referred to as LFDnPatch, as it is a learning-based light field denoising method relying on a patch-matching algorithm and inspired by its image denoising counterpart, i.e. DnCNN.[18]

4.1 LFDnPatch

As a preliminary step, it is necessary to fix the disparity problem. The simplest solution would be to shift the neighboring images of every SAI so they fit better. However, in practice, this shift is not constant throughout the image and the implementation of such a method would be most likely inaccurate and computationally expensive.

It is instead possible to construct images similar to a given SAI using the remaining images in the scene. This idea was first analyzed in the work of Fan *et al.*[24] on light field super-resolution, where a set of stitch-patched versions of the image is produced and used to give extra information to the CNN.

The method proposed in this paper takes an entire grid of SAIs as input. For each SAI, N stitch-patched images ($N = 6$ in our implementation) are built by decomposing each SAI into non-overlapping patches and then finding similar patches in the other SAIs, where the similarity is computed via mean absolute error. The first stitch-patched image is built by using the most similar patch found, the second image using the second

most similar, and so on. This way, one obtains several images that are locally similar to the original, and result from stitching all these patches together. The images will never be identical, because of disparities and occlusions introduced by the different viewpoints, but by making them locally similar we make it easier to exploit redundancies for denoising.

Two examples of such stitch-patched versions are illustrated in Figure 2. The top images were taken from a scene with very large angular spacing between SAIs, which leads to the very visible blocking artifacts on the stitch-patched version. This happens because the large angular spacing means that there are a lot of occlusions and disparity effects, and so while the searched patches are similar to the reference, they present visual discontinuities when stitched together. On the other hand, in the images at the bottom, it is much harder to spot such artifacts, which suggests that the patch-matching method gives adequate results under smaller angular spacing.

The built images are stacked along the third dimension, producing an array of shape $[H, W, (N+1) \cdot C]$, where C is 3 for RGB images and 1 for grayscale. The last C channels correspond to the reference image, i.e. the image used to build all the stitch-patched versions. This augmented image is fed into a network having a similar architecture to the DnCNN, where the first convolutional layer has been adjusted to handle such input. The network was then trained to produce a noise map, which is then subtracted from the original noisy image to obtain the denoised estimate.

4.1.1 Patch match algorithm

It is worthwhile describing the patch-matching algorithm in more detail, as several heuristics had to be used to reduce its run time. Considering an image grid of shape $[G_H, G_W]$, composed of SAIs of shape $[H, W]$, the goal is to compose N patched-stitched versions of the SAI in position (i, j) in the grid, taking patches of dimensions $[k, k]$.

An optimal implementation of the patch-matching algorithm would go through every other SAI in the scene, and within every SAI it would look at every possible patch in the scene in search of the best candidates. While this method would yield the best results, it is also prohibitively expensive. The number of operations to compare one patch with another is k^2, and each SAI has $\lceil H/k \rceil \cdot \lceil W/k \rceil$ non-overlapping patches. Additionally, considering that each reference patch from each SAI needs to be compared with every one of the $(H-k+1) \cdot (W-k+1)$ overlapping patches in all the other SAIs and that there are $G_H \cdot G_W$ SAIs, this results in a total complexity of

$$\mathcal{O}(H^2 \cdot W^2 \cdot G_H^2 \cdot G_W^2)$$

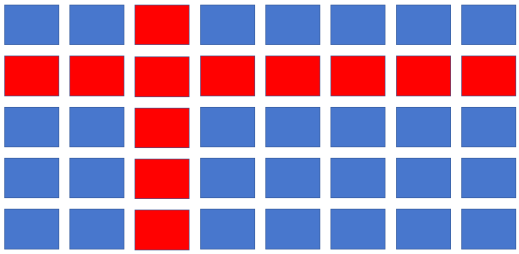
Considering that it is still necessary to sort the candidates (which takes roughly $\mathcal{O}(H^2 \cdot W^2 \cdot G_H^2 \cdot G_W^2 \cdot k^{-2} \cdot \log N)$ if done with binary insertion), it is evident that such an approach cannot scale. So, several heuristics were used to speed up the process, as represented in Figure 3.

The first heuristic stems from the fact that there are strong geometrical constraints that can be taken into account within an SAI grid. Namely, it can be assumed that for SAIs in the same row of the grid, there is only horizontal disparity. Likewise, for images in the same column of the grid, we can assume there is only vertical disparity. This means that for these SAIs, one only needs to search in one direction of the image relative to the current patch, and not in the entire image, as shown in Figure 3b, hence significantly reducing the computational cost. This does not exclude the presence of viable, or even better candidates in the remaining images of the grid, but they would be much more costly to find. Figure 3a represents an SAI grid, where, when looking for patch matches for the SAI in the center of the cross, only the red images are considered in the search. This heuristic already simplifies the complexity to

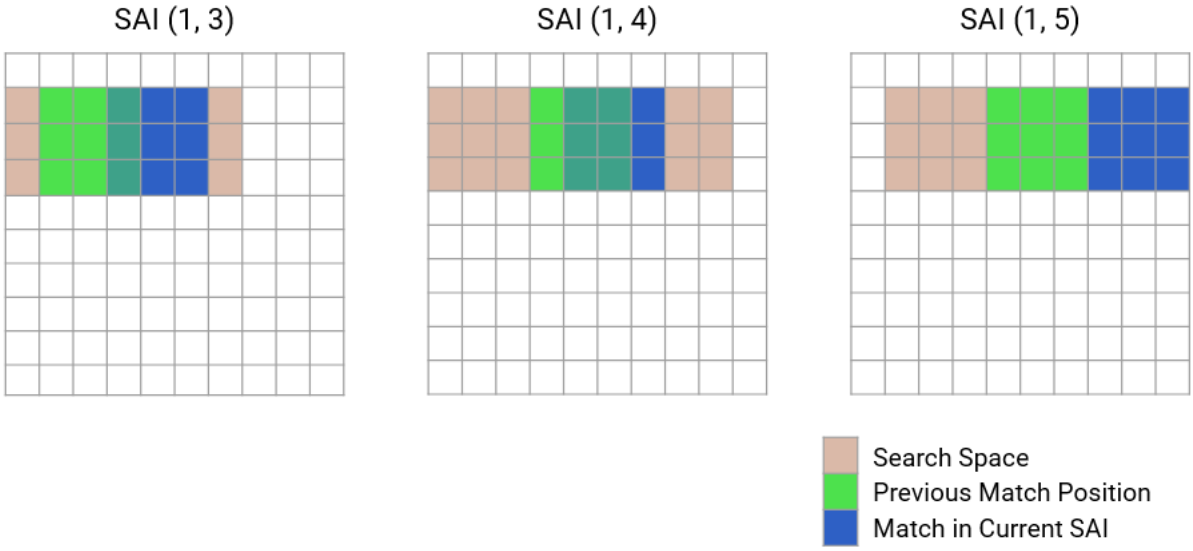
$$\mathcal{O}(G_H \cdot G_W \cdot H \cdot W \cdot (H \cdot G_H + W \cdot G_W))$$

Having narrowed the search to a 1D problem instead of a 2D problem, it is also possible to simplify it further by assuming that the disparity between neighboring SAIs is not large. This means that it is only necessary to search in the vicinity of the position of the closest match in the previous SAI. This is essentially a sliding window, represented in Figure 3b by the region in brown. If there are $S \ll H, W$ positions within the search space, then the final complexity for the simplified algorithm becomes

$$\mathcal{O}(G_H \cdot G_W \cdot H \cdot W \cdot S \cdot (G_H + G_W)) \tag{1}$$



(a) In red are the only SAIs considered when applying patch-matching to the one at the intersection. All the blue ones are ignored.



(b) Representation of the 1D search for the closest match over a row of SAIs.

Figure 3: Visual representation of the heuristics applied to speed up the patch-matching algorithm

In comparison with the naïve implementation's complexity, there is a tremendous gain, that becomes even more significant as the images/scenes get larger.

4.2 Training routine

As the process of computing the stitch-patched versions is too slow to be performed at every step of training, they were computed for the entire training dataset beforehand and used as the input to train the model. During the training procedure, the patches were computed on the clean images, and then random gaussian noise was applied to the stitch-patched versions at each step of training. Nevertheless, in reality, patch-matching would be performed on noisy images in order to introduce differences between the training pipeline and the inference pipeline.

To train the model, we used l_2 loss, as well as a step-based decaying learning rate. We used a batch size of 128 and a training patch of shape $[21, 64, 64]$, 21 channels from the stitch-patched images, and square patches with side 64 pixels. Using 2 GPUs, the training took 20 hours for the model, as well as 100 hours (on CPU) to compute the stitch-patched versions for all images.

To train the model for blind denoising, the noise strength was not fixed during training, i.e. for each of the extracted training patches, a random Gaussian noise strength was selected in the range [0-55], and noise with such strength was applied to the patch.

Parameter	a	b	c	d
Best Fit	4.8×10^{-3}	2.0	6.1×10^{-1}	3.4

Table 2: Best-fit parameters from Equation 2 to data for LFDnPatch in Figure 4.

Parameter	a	b	c	d
Best Fit	1.3×10^{-3}	1.9	3.2×10^{-1}	0.0

Table 3: Best-fit parameters from Equation 2 to data for LFDnPatch in Figure 5.

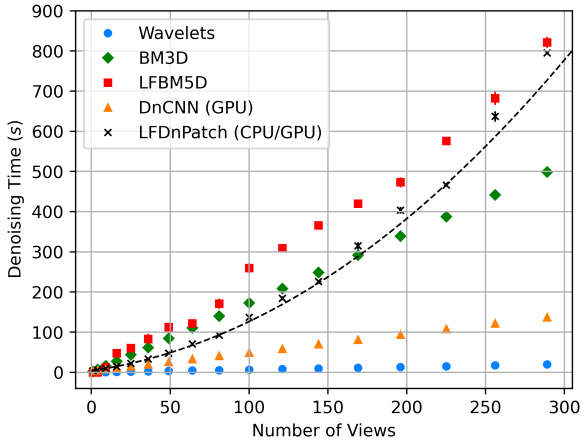


Figure 4: Execution times for the methods as a function of scene size. The dashed line corresponds to the fit of Equation 2 to the LFDnPatch data points.

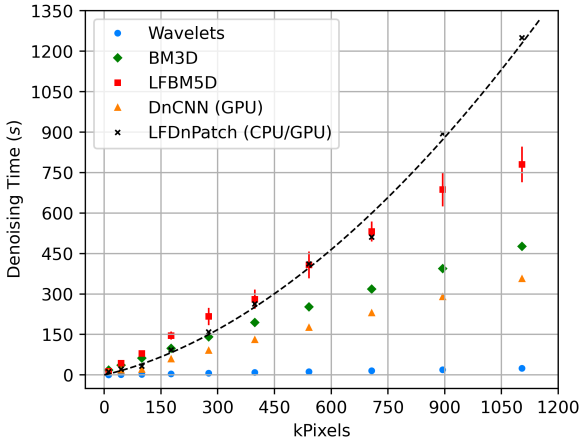


Figure 5: Execution times for the methods as a function of SAI size. The dashed line corresponds to the fit of Equation 2 to the LFDnPatch data points.

5. EMPIRICAL COMPLEXITY ANALYSIS

In this section, the empirical effect of scene size (number of views per scene) and view size (number of pixels) on the run-time of the considered methods are reviewed.

5.1 Hardware specifications

For the experiments, two different hardware were used. Notably, LFDnPatch runs on a hybrid set-up where the construction of the stitch-patched versions is performed on an Intel Xeon CPU E5-2630 CPU with 20 cores, and the processing of the enhanced image by the learned model is performed on an Nvidia GeForce RTX 2080 GPU.

5.2 Scene size effect on run-time

To assess how the scene size affects the run-time, the scene `flower` was used with a downscaling factor of 0.3. This choice is motivated by our sole interest in studying how the scene size affects the execution time of each method. A smaller image on all tests allows running the tests faster without impacting the results.

The results of the proposed method have been compared to:

- Two conventional methods for image denoising, i.e. Wavelet thresholding and BM3D.[37]
- One learning-based method for image denoising, i.e. DnCNN.[18]
- One conventional method for light field denoising, i.e. LFBM5D.[22]

From an implementation perspective, for Wavelet thresholding the implementation from SciPy[38] was used, which implements Wavelet denoising via adaptive thresholding by computing separate thresholds for each wavelet sub-band, instead of a universal threshold, as introduced by Chang *et al.*[39] For BM3D, the implementation submitted alongside the LFBM5D[40] was used for the experiments. This was because using the codes for BM3D and LFBM5D with exactly the same compilation settings and consistent mode of usage makes it fairer to

compare them in terms of run-time complexity. Finally, for DnCNN, the original implementation[41] provided by the authors was used.

The results are presented in Figure 4. The original `flower` scene had a scene size of (17×17), leading to a total of 289 SAIs. The different-sized scenes were extracted from the original using sub-grids of size $(i \times i), i \in [1, 17]$ starting from the top-left corner of the scene.

Considering that the methods BM3D, Wavelets, and DnCNN were initially designed to denoise images, it is expected to see their run-time grow linearly with the scene size. Indeed that is observed, with BM3D being the slowest and Wavelets the fastest. However, it should be pointed out that the DnCNN method is running on GPU, not CPU, which gives it its boost over BM3D. While the LFBM5D is a light field-specific method, it also seems to grow linearly with scene size. This results from trade-offs of speed and performance implemented by the authors, as a naive implementation of LFBM5D would not result in linear complexity.

For LFDnPatch, a more careful analysis should be performed. One can observe that it is growing superlinearly, which is expected as Equation 1 dictates a dependency of the form $\mathcal{O}(G_H \cdot G_W \cdot (G_H + G_W))$. For simplicity, let us take $G \coloneqq G_H \approx G_W$ (exactly true in our experiments). So, if there are $N \coloneqq G^2$ images in the scene, the dependency should be of the form $\mathcal{O}(N^{3/2})$. To test this, we fit a function of the form

$$aN^b + cN + d \tag{2}$$

with free parameters a, b, c, d. The linear term is the contribution of running the stitch-patched images through the model, which is linear. The result of the fit is shown in Figure 4 as the black dashed line, and the best-fit parameters are in Table 2. The most important parameter is $b = 2.0$, which is slightly higher than the theoretical optimum of 1.5. This suggests that the code can be further optimized in terms of scaling with scene size.

5.3 SAI size effect on run-time

Similarly to the previous section, the effect of the SAI size affects the run-time of each method was also studied. The `flower` scene was used again, but to speed up the tests only used a sub-grid of size (6×6) was considered. The results are presented in Figure 5.

It can be observed that the methods Wavelets, DnCNN, BM3D, and LFBM5D also grow linearly with the number of pixels. For the Wavelets and DnCNN, this happens by design, while for the BM3D and LFBM5D, it is due to internal heuristics that speed up the implementation, making it linear with image size. Additionally, one can observe that LFBM5D has a much higher variance in its run-time when compared to other methods.

Regarding LFDnPatch, it should behave as $\mathcal{O}(H \cdot W)$, i.e. linear with image size. However, the data in the figure does not match the latter. Fitting again Equation 2 one obtains the parameters in Table 3. The most relevant parameter is again $b = 1.9$ and as it can be seen, it is far from the expected 1. This is another strong indicator that the code can be further optimized, probably with regard to memory allocations or data movements.

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Objective results

To perform a quantitative comparison of all the methods, the scenes `backlight`, `pens`, `fence`, `t-rex`, and `reflection` from the test set were used. For each scene, either additive white Gaussian or Poissonian-Gaussian noise were applied, with three increasing strength levels, as can be seen in Figure 6. Then all the denoising methods were applied to these scenes and the reconstruction quality was measured using the PSNR and SSIM objective quality metrics. The full results can be found in Appendix A, respectively in Tables 5 and 6. It should be noted that BM3D and LFBM5D require an estimate of the noise variance for parameter selection. Noise variance as a strength indicator was also used in the analysis. To estimate the noise variance the function `estimate_sigma` from the scikit-image package[42] was used, which implements the method proposed by Donoho and Johnstone.[43] However, this method is designed for additive white gaussian noise, which is not a realistic noise. Therefore, we will make a distinction between using σ for gaussian noise and σ_{est} for realistic noise.

(a) Soft Gaussian noise ($\sigma = 5$). (b) Medium Gaussian noise ($\sigma = 13$). (c) Hard Gaussian noise ($\sigma = 20$).

(d) Soft Poissonian-Gaussian noise ($\sigma_{est} = 6$). (e) Medium Poissonian-Gaussian noise ($\sigma_{est} = 13$). (f) Hard Poissonian-Gaussian noise ($\sigma_{est} = 20$).

Figure 6: Examples of noisy images, for each of the strengths and noise types considered.

Table 4: Aggregated results for additive white gaussian noise and realistic noise. The aggregation is done via averaging over Tables 5 and 6 for each method.

(a) Aggregated method results for additive white gaussian noise.

Method	Soft Noise		Medium Noise		Hard Noise	
	$\overline{\text{PSNR}}$	$\overline{\text{SSIM}}$	$\overline{\text{PSNR}}$	$\overline{\text{SSIM}}$	$\overline{\text{PSNR}}$	$\overline{\text{SSIM}}$
Wavelets	39.3	0.936	32.1	0.767	29.6	0.688
BM3D	42.5	0.961	34.1	0.823	31.3	0.751
LFBM5D	**43.1**	**0.965**	**34.9**	**0.859**	**31.9**	**0.805**
DnCNN	41.2	0.957	34.6	0.853	**31.8**	0.796
LFDnPatch	40.4	0.951	33.9	0.833	31.2	0.773

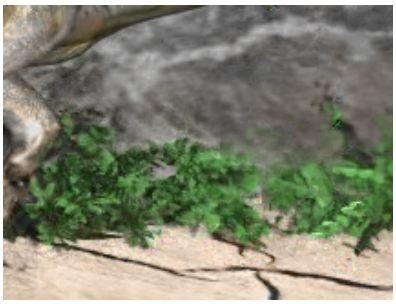
(b) Aggregated method results for realistic noise.

Method	Soft Noise		Medium Noise		Hard Noise	
	$\overline{\text{PSNR}}$	$\overline{\text{SSIM}}$	$\overline{\text{PSNR}}$	$\overline{\text{SSIM}}$	$\overline{\text{PSNR}}$	$\overline{\text{SSIM}}$
Wavelets	38.4	0.904	32.2	0.768	29.0	0.654
BM3D	40.1	0.927	33.5	0.804	30.0	0.669
LFBM5D	**41.0**	0.942	**35.0**	**0.857**	**30.3**	**0.755**
DnCNN	40.4	**0.945**	34.5	**0.857**	**30.5**	0.748
LFDnPatch	39.5	0.935	33.8	0.820	29.7	0.717

Table 4 shows the aggregated results, where one can see that LFBM5D presented the highest performance, consistently outscoring the other methods both in terms of SSIM and PSNR scores. This is a strong indication that light field-specific methods are capable of exploiting scene redundancies and achieving better performance

Clean patch.

BM3D output.
(PSNR=28.2; SSIM=0.857)



LFBM5D output.
(PSNR=26.5 SSIM=0.818)

Noisy patch.
(PSNR=23.0; SSIM=0.530)

DnCNN output.
(PSNR=28.1; SSIM=0.860)

LFDnPatch output.
(PSNR=26.6; SSIM=0.791)

Figure 7: Detail from `t-rex` scene, clean, noisy (gaussian hard noise) and as the output of denoising methods.

than standard image denoising methods applied to the different SAIs. However, one interesting takeaway is that, looking at the full results in Appendix A, LFBM5D presents lower performance than BM3D or DnCNN in the `t-rex` scene. This matches our expectations, as this scene presents particularly large angular spacing. In this way, the redundancy between the different SAIs is reduced and more occlusions and disparity effects make light field-specific methods inefficient. A similar behavior is, in fact, also presented for the LFDnPatch method for the same scene. This suggests that specific methods for light fields with large angular should be designed to mitigate this problem.

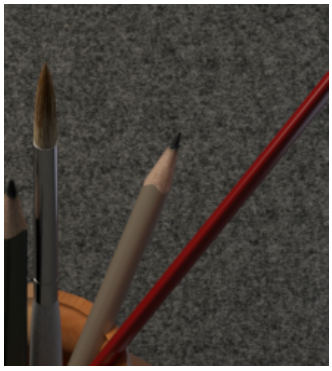
Another observation is that the BM3D method slightly outperforms DnCNN for soft noise, but as the noise intensity increases the DnCNN becomes more competitive and surpasses BM3D, even becoming comparable to LFBM5D. This shows that learning-based methods are capable of adapting better than classical methods to stronger noise levels. Adding this to the fact that these methods tend to be faster, applications with strong noise and strong time constraints may benefit from learning-based methods.

Another observation regards the fact that only a small performance drop can be observed between the additive white gaussian noise and the Poissonian-Gaussian noise. This stems from the fact that while these methods were designed/trained for additive white gaussian noise, they are capable of generalizing to other noise statistics without being optimized for them.

Looking at LFDnPatch, one observes that even though it performed slightly worse than DnCNN, it always performed better than Wavelet denoising, and outperforms BM3D in the case of medium and strong noise. The results are promising, and it seems feasible that this method will reach comparable or even superior performance when compared to DnCNN, in the future. Finally, one must remember that IQA metrics aren't perfectly correlated with human perception, and are therefore not sufficient for a complete analysis.

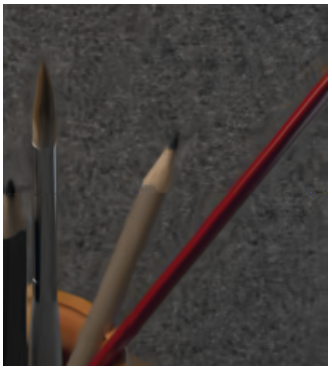
6.2 Visual examples

To provide a visual assessment of the methods, a number of patches from the testing scenes are presented. For each, the reference-clean patch, the noisy patch, and the outputs from the methods BM3D, LFBM5D, DnCNN, and LFDnPatch are shown. The Wavelet Denoising method was discarded in this analysis.

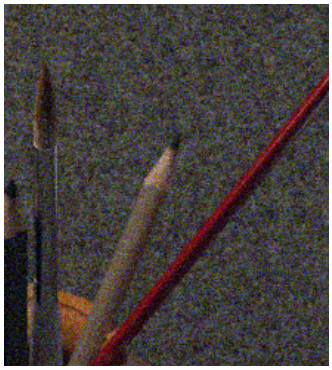


Clean patch.

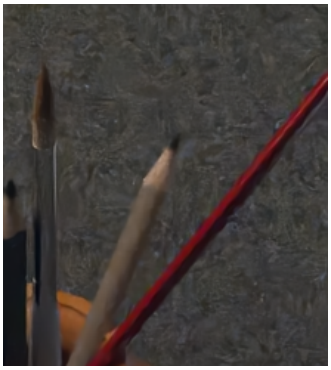
BM3D output.
(PSNR=31.7; SSIM=0.732)



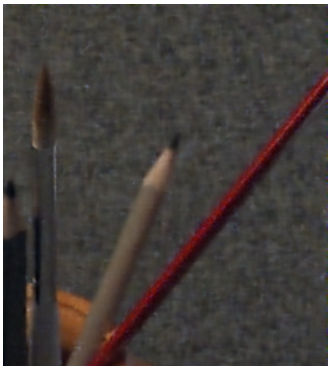
LFBM5D output.
(PSNR=33.0; SSIM=0.825)



Noisy patch.
(PSNR=24.9; SSIM=0.353)



DnCNN output.
(PSNR=31.5; SSIM=0.753)



LFDnPatch output.
(PSNR=31.5; SSIM=0.741)

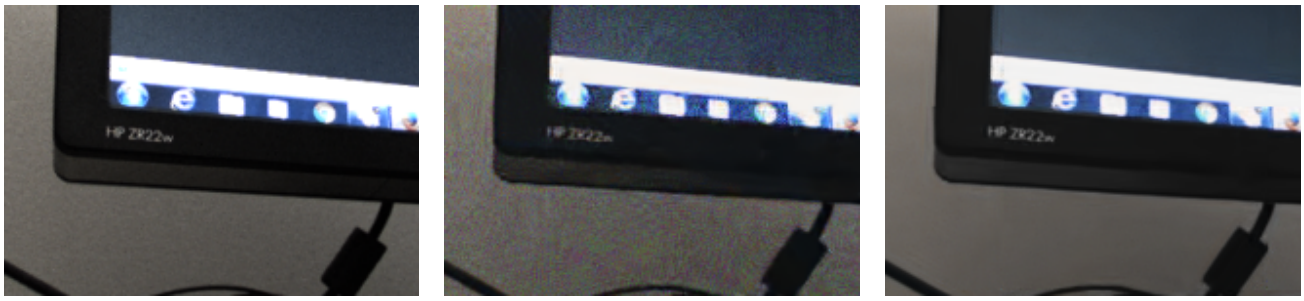
Figure 8: Detail from `translucent` scene, clean, noisy (realistic hard) and as output of denoising methods.

Figure 7 reports a crop from the `t-rex` scene. This is the scene presenting the largest angular disparity, and it can be observed that the performance for light field-specific denoising methods is lower than the image-based denoisers. Looking at their behaviors in the fine structures of the foliage, it is possible to observe that the output from LFBM5D and LFDnPatch is more blurry, even erasing certain fine details.

Figure 8 shows another challenging scene, as the background has a texture very similar to noise. Due to this fact, all the methods produced blurry output for the background, as they were not capable of distinguishing it from noise. However, the LFBM5D method is the only method that kept the sharpness on the foreground objects, which is most clearly seen in the red pencil. In this scene, the LFDnPatch method presented the lowest performance as, while the noise was effectively removed, the output presents high blurriness. The BM3D and DnCNN, on the other hand, present comparable performances.

Figure 9 shows how the methods generalize in the case of realistic noise. The noisy patch shows the predominance of noise in the green channel compared to the red and blue. From the output of BM3D, it is possible to observe that this method is not able to effectively remove the noise in the green channel. This stems from the fact that this method was designed for additive white gaussian noise, and so expects similar noise strengths along the channels. Comparing it with DnCNN, it seems that learning-based methods may generalize better to different noise statistics, as they were not designed with specific priors in mind.The LFBM5D method, on the other hand, seems to exploit the light field redundancies in a way that allows it to overcome its design priors, giving an image almost identical to the original, when the results are assessed visually.

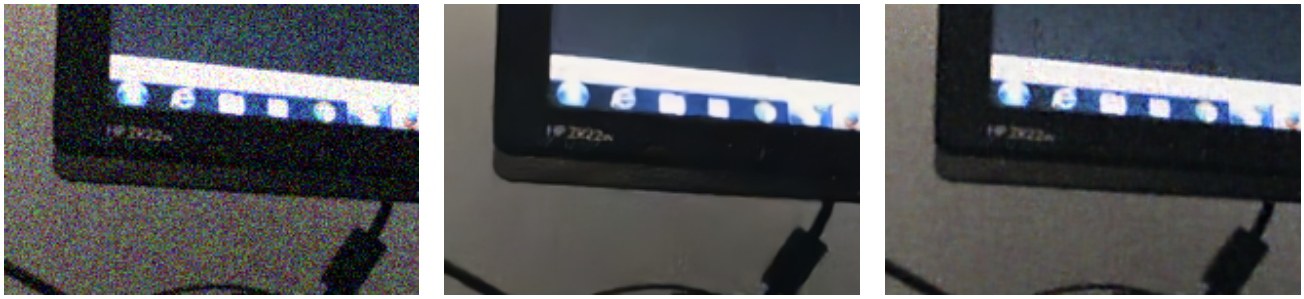
Finally, Figure 10 shows one last example. Here it is clear that LFBM5D and DnCNN present the highest performance, but it is interesting how BM3D and LFDnPatch seem to behave in different ways. The results from BM3D seem grainy but they still retain the sharpness of the original image. On the other hand, the results from LFDnPatch are better at restoring the original textures but introduce more blur to the image.



Clean patch.

BM3D output.
(PSNR=29.8; SSIM=0.696)

LFBM5D output.
(PSNR=32.2; SSIM=0.904)



Noisy patch.
(PSNR=21.6; SSIM=0.256)

DnCNN output.
(PSNR=31.9; SSIM=0.896)

LFDnPatch output.
(PSNR=30.2; SSIM=0.833)

Figure 9: Detail from `backlight` scene, clean, noisy (realistic hard) and as output of denoising methods.

Clean patch.

BM3D output.
(PSNR=32.3; SSIM=0.681)

LFBM5D output.
(PSNR=33.5; SSIM=0.733)

Noisy patch.
(PSNR=26.5; SSIM=0.329)

DnCNN output.
(PSNR=33.4; SSIM=0.736)

LFDnPatch output.
(PSNR=32.9; SSIM=0.719)

Figure 10: Detail from `fence` scene, clean, noisy (gaussian medium) and as output of denoising methods.

These results visually confirm that the method that presents the highest performance, both visually and in terms of objective scores, is LFBM5D, except for images with large angular spacing. Nevertheless, all the

methods present comparable results, with BM3D and LFDnPatch performing slightly worse than DnCNN. Since both DnCNN and LFBM5D outperformed BM3D, we believe that this provides evidence that a learning-based light field-specific method such as LFDnPatch can further improve light field denoising performance.

7. CONCLUSIONS

In this paper, a novel learning-based light field denoising method was presented. While assessing its performance, several existing image and light field denoising methods have been reviewed. A large dataset of light fields was also built by merging multiple noise-free scenes from existing datasets and applying synthetic noise. The comparison of BM3D, i.e. a classical image denoising method, with LFBM5D, i.e. its light field denoising generalization, showed that while there is a clear performance boost by using the latter, it is also computationally heavier, as the added dimensions increase its run-time complexity. However, for scenes with large angular spacings, none of the light field denoising methods were able to outperform their image denoising counterparts. The proposed novel method is based on a patch-matching algorithm to build stitch-patched versions of every SAI from its neighboring SAIs. This allows us to exploit redundancies in similar patches as an input to a denoising learning-based model. Objective results show that despite the fact that the proposed method presented lower performance compared to the original DnCNN, it is capable of exploiting the information over the different views through the stitch-patched images and generalizing across the noise types. The results presented in this paper are only preliminary and several directions still need to be explored. In future work, different architectures may be explored, e.g. by studying a 2-stage model.

ACKNOWLEDGMENTS

The authors would like to acknowledge support from the Swiss National Scientific Research project entitled "Compression of Visual information for Humans and Machines (CoViHM)" under grant number 200020_207918.

REFERENCES

[1] Buades, A., Coll, B., and Morel, J.-M., "A non-local algorithm for image denoising," in [*2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*], **2**, 60–65 vol. 2 (2005).

[2] Xu, J., Zhang, L., Zuo, W., Zhang, D., and Feng, X., "Patch group based nonlocal self-similarity prior learning for image denoising," in [*2015 IEEE International Conference on Computer Vision (ICCV)*], 244–252 (2015).

[3] Elad, M. and Aharon, M., "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing* **15**(12), 3736–3745 (2006).

[4] Zha, Z., Liu, X., Huang, X., Shi, H., Xu, Y., Wang, Q., Tang, L., and Zhang, X., "Analyzing the group sparsity based on the rank minimization methods," in [*2017 IEEE International Conference on Multimedia and Expo, ICME 2017, Hong Kong, China, July 10-14, 2017*], 883–888, IEEE Computer Society (2017).

[5] Rudin, L., Osher, S., and Fatemi, E., "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena* **60**, 259–268 (November 1992).

[6] Weiss, Y. and Freeman, W. T., "What makes a good model of natural images?," in [*2007 IEEE Conference on Computer Vision and Pattern Recognition*], 1–8 (2007).

[7] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K., "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing* **16**(8), 2080–2095 (2007).

[8] Gu, S., Xie, Q., Meng, D., Zuo, W., Feng, X., and Zhang, L., "Weighted nuclear norm minimization and its applications to low level vision," *International Journal of Computer Vision* **121** (January 2017).

[9] Schmidt, U. and Roth, S., "Shrinkage fields for effective image restoration," in [*2014 IEEE Conference on Computer Vision and Pattern Recognition*], 2774–2781 (2014).

[10] Chen, Y., Yu, W., and Pock, T., "On learning optimized reaction diffusion processes for effective image restoration," in [*IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*], 5261–5269, IEEE Computer Society (2015).

[11] Chen, Y. and Pock, T., "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**(6), 1256–1272 (2017).

[12] Pardasani, R. and Shreemali, U., "Image denoising and super-resolution using residual learning of deep convolutional network," (2018).

[13] Tian, C., Xu, Y., Zuo, W., Du, B., Lin, C.-W., and Zhang, D., "Designing and training of a dual cnn for image denoising," *Knowledge-Based Systems* **226**, 106949 (2021).

[14] Xiao, P., Guo, Y., and Zhuang, P., "Removing stripe noise from infrared cloud images via deep convolutional networks," *IEEE Photonics Journal* **10**(4), 1–14 (2018).

[15] Tripathi, S., Lipton, Z. C., and Nguyen, T. Q., "Correction by projection: Denoising images with generative adversarial networks," (2018).

[16] Yu, A., Liu, X., Wei, X., Fu, T., and Liu, D., "Generative adversarial networks with dense connection for optical coherence tomography images denoising," in [*2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*], 1–5 (2018).

[17] Tian, C., Fei, L., Zheng, W., Xu, Y., Zuo, W., and Lin, C.-W., "Deep learning on image denoising: An overview," (2019).

[18] Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L., "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing* **26**(7), 3142–3155 (2017).

[19] Li, Z., Baker, H., and Bajcsy, R., "Joint image denoising using light-field data," in [*2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*], 1–6 (2013).

[20] Mitra, K. and Veeraraghavan, A., "Light field denoising, light field superresolution and stereo camera based refocussing using a gmm light field patch prior," in [*2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*], 22–28 (2012).

[21] Sepas-Moghaddam, A., Correia, P. L., and Pereira, F., "Light field denoising: exploiting the redundancy of an epipolar sequence representation," in [*2016 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*], 1–4 (2016).

[22] Alain, M. and Smolic, A., "Light field denoising by sparse 5d transform domain collaborative filtering," in [*2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*], 1–6 (2017).

[23] Chen, J., Hou, J., and Chau, L., "Light Field Denoising via Anisotropic Parallax Analysis in a CNN Framework," *IEEE Signal Process. Lett.* **25**(9), 1403–1407 (2018).

[24] Fan, H., Liu, D., Xiong, Z., and Wu, F., "Two-stage convolutional neural network for light field super-resolution," in [*2017 IEEE International Conference on Image Processing (ICIP)*], 1167–1171 (2017).

[25] Dansereau, D. G., Pizarro, O., and Williams, S. B., "Decoding, Calibration and Rectification for lenselet-Based Plenoptic Cameras," in [*Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*], IEEE (June 2013).

[26] Hahne, C. and Aggoun, A., "PlenoptiCam v1.0: A Light-Field Imaging Framework," *IEEE Transactions on Image Processing* **30**, 6757–6771 (2021).

[27] Raj, A. S., Lowney, M., Shah, R., and Wetzstein, G., "Stanford Lytro Light Field Archive," (2016). [ONLINE] Available on: http://lightfields.stanford.edu/LF2016.html.

[28] Shi, J., Jiang, X., and Guillemot, C., "A Framework for Learning Depth From a Flexible Subset of Dense and Sparse Light Field Views," *IEEE Transactions on Image Processing* **28**(12), 5867–5880 (2019). [ONLINE] Available on: http://clim.inria.fr/Datasets/InriaSynLF/index.html.

[29] Rerábek, M. and Ebrahimi, T., "New Light Field Image Dataset," in [*International Workshop on Quality of Multimedia Experience*], (2016). [ONLINE] Available on: https://www.epfl.ch/labs/mmspg/downloads/epfl-light-field-image-dataset/.

[30] Honauer, K., Johannsen, O., Kondermann, D., and Goldluecke, B., "A Dataset and Evaluation Methodology for Depth Estimation on 4D Light Fields," in [*Computer Vision – ACCV 2016*], 19–34, Springer International Publishing, Cham (2017). [ONLINE] Available on: https://lightfield-analysis.uni-konstanz.de.

[31] Johannsen, O., Honauer, K., Goldluecke, B., Alperovich, A., Battisti, F., Bok, Y., Brizzi, M., Carli, M., Choe, G., Diebold, M., Gutsche, M., Jeon, H.-G., Kweon, I. S., Neri, A., Park, J., Park, J., Schilling, H., Sheng, H., Si, L., Strecke, M., Sulc, A., Tai, Y.-W., Wang, Q., Wang, T.-C., Wanner, S., Xiong, Z., Yu, J., Zhang, S., and Zhu, H., "A taxonomy and evaluation of dense light field depth estimation algorithms," in [*Conference on Computer Vision and Pattern Recognition - LF4CV Workshop*], (2017).

[32] Stanford Computer Graphics Laboratory, "Stanford Light Field Archive," (2008). [ONLINE] Available on: http://lightfield.stanford.edu/lfs.html.

[33] Wetzstein, G., "Synthetic Light Field Archive," (2013). [ONLINE] Available on: https://web.media.mit.edu/~gordonw/SyntheticLightFields.

[34] Marwah, K., Wetzstein, G., Veeraraghavan, A., and Raskar, R., "Compressive light field photography," (August 2012).

[35] Wetzstein, G., Lanman, D., Hirsch, M., and Raskar, R., "Tensor displays: compressive light field synthesis using multilayer displays with directional backlighting," *ACM Transactions on Graphics* **31**, 1–11 (July 2012).

[36] Wetzstein, G., Lanman, D., Heidrich, W., and Raskar, R., "Layered 3d: Tomographic image synthesis for attenuation-based light field and high dynamic range displays," *ACM Trans. Graph.* **30**, 95 (July 2011).

[37] Ymir Mäkinen, Lucio Azzari, A. F., "Wrapper for BM3D denoising," (October 2021). [ONLINE] Available on: https://webpages.tuni.fi/foi/GCF-BM3D/.

[38] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods* **17**, 261–272 (2020).

[39] Chang, S., Yu, B., and Vetterli, M., "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing* **9**(9), 1532–1546 (2000).

[40] Alain, M., "C/C++ implementation of the LFBM5D filter for light field denoising and super-resolution," (April 2020). [ONLINE] Available on: https://github.com/V-Sense/LFBM5D.

[41] Zhang, K., "Image Restoration Toolbox (PyTorch). Training and testing codes for DPIR, USRNet, DnCNN, FFDNet, SRMD, DPSR, BSRGAN, SwinIR," (September 2021). [ONLINE] Available on: https://github.com/cszn/KAIR/.

[42] van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ* **2**, e453 (June 2014).

[43] Donoho, D. L. and Johnstone, I. M., "Ideal spatial adaptation by wavelet shrinkage," *Biometrika* **81**, 425–455 (September 1994).

APPENDIX A. COMPLETE SCENE SCORES

In this section, we include the full results obtained while comparing different methods. Table 5 shows the results from additive gaussian noise, with soft ($\sigma = 5$), medium ($\sigma = 13$), and hard ($\sigma = 20$) noise. Likewise, Table 6 shows the results from realistic noise, with soft ($\sigma_{\text{est}} = 6$), medium ($\sigma_{\text{est}} = 13$), and hard ($\sigma_{\text{est}} = 20$) noise.

Table 5: Results for all methods applied on the selected testing scenes, with additive white gaussian noise.

Method	backlight		pens		fence		t-rex		reflection	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$\sigma = 5$										
Noisy	36.3	0.810	36.7	0.862	36.4	0.759	36.4	0.963	36.8	0.732
Wavelets	38.6	0.934	37.3	0.930	39.4	0.902	39.4	0.979	41.7	0.937
BM3D	**41.6**	**0.970**	40.8	0.969	42.4	0.915	**41.3**	**0.987**	46.5	**0.966**
LFBM5D	**41.5**	**0.968**	**42.2**	**0.980**	**44.0**	**0.924**	40.6	**0.985**	**47.2**	**0.967**
DnCNN	40.6	0.965	40.3	0.968	42.1	0.909	38.4	0.980	44.8	0.961
LFDnPatch	40.0	0.955	39.4	0.954	41.6	0.917	37.5	0.973	43.6	0.956
$\sigma = 13$										
Noisy	27.3	0.409	28.4	0.507	26.6	0.282	29.0	0.738	26.7	0.229
Wavelets	32.0	0.798	32.6	0.799	31.2	0.605	31.0	0.885	33.8	0.746
BM3D	34.8	0.888	35.1	0.885	32.4	0.632	**33.6**	**0.938**	34.7	0.770
LFBM5D	**35.6**	**0.923**	**37.1**	**0.933**	**33.5**	**0.699**	32.8	0.928	35.3	0.812
DnCNN	35.3	0.913	35.2	0.892	**33.6**	**0.699**	33.1	0.936	**35.8**	**0.824**
LFDnPatch	34.4	0.895	34.5	0.862	33.2	0.685	32.0	0.912	35.4	0.810
$\sigma = 20$										
Noisy	23.8	0.266	24.4	0.307	24.3	0.198	24.5	0.550	24.6	0.159
Wavelets	29.1	0.714	30.7	0.710	29.2	0.532	27.6	0.798	31.4	0.687
BM3D	31.2	0.797	32.9	0.811	30.0	0.556	**30.4**	**0.889**	31.9	0.702
LFBM5D	**31.9**	**0.875**	**34.8**	**0.879**	**30.9**	**0.639**	29.6	0.874	32.5	0.759
DnCNN	**31.9**	0.865	33.1	0.825	**31.0**	0.634	30.3	**0.890**	**32.8**	**0.768**
LFDnPatch	31.0	0.835	32.8	0.800	30.6	0.616	29.2	0.856	32.4	0.759

Table 6: Results for all methods applied on the selected testing scenes, with realistic noise.

Method	backlight		pens		fence		t-rex		reflection	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$\sigma_{\text{est}} = 6$										
Noisy	33.9	0.760	32.5	0.799	38.0	0.697	39.2	0.875	37.2	0.668
Wavelets	37.0	0.914	35.2	0.890	39.4	0.841	**38.6**	0.960	41.8	0.915
BM3D	40.0	0.959	38.8	0.955	40.1	0.831	37.4	**0.965**	44.3	0.924
LFBM5D	**40.5**	**0.966**	**41.5**	**0.974**	41.4	0.863	36.4	0.960	**45.4**	0.945
DnCNN	39.8	0.960	38.9	0.954	**42.0**	0.898	35.9	0.960	45.2	**0.954**
LFDnPatch	39.0	0.949	37.7	0.937	41.0	**0.901**	35.9	0.941	43.9	0.946
$\sigma_{\text{est}} = 13$										
Noisy	27.1	0.473	26.4	0.500	30.5	0.342	30.8	0.737	24.6	0.307
Wavelets	32.1	0.798	31.8	0.767	33.8	0.636	31.2	0.883	31.9	0.754
BM3D	34.4	0.866	34.6	0.876	34.3	0.616	**32.6**	0.914	31.6	0.750
LFBM5D	**36.0**	**0.926**	**37.3**	**0.932**	**36.0**	0.695	32.3	0.916	33.3	0.817
DnCNN	35.6	0.914	35.0	0.886	35.7	**0.718**	32.4	**0.923**	**33.8**	**0.844**
LFDnPatch	34.6	0.895	34.3	0.855	34.8	0.701	31.5	0.896	**33.9**	0.754
$\sigma_{\text{est}} = 20$										
Noisy	22.2	0.224	25.6	0.325	24.7	0.154	26.0	0.591	23.9	0.103
Wavelets	27.8	0.670	31.2	0.729	28.9	0.481	27.1	0.778	29.8	0.610
BM3D	29.4	0.738	32.9	0.777	29.5	0.463	28.2	0.775	30.0	0.594
LFBM5D	**30.2**	**0.848**	**34.4**	**0.858**	**30.0**	**0.568**	28.4	0.832	28.7	0.667
DnCNN	**30.2**	0.835	32.7	0.798	**30.0**	**0.570**	**29.0**	**0.853**	**30.6**	**0.686**
LFDnPatch	28.5	0.795	32.7	0.783	29.3	0.544	28.0	0.812	30.0	0.650