Supplementary Material

Shachar Ben Dayan
bendayan@mail.tau.ac.il
David Mendlovic
mend@eng.tau.ac.il
Raja Giryes
raja@tauex.tau.ac.il

School of Electrical Engineering
The Iby and Aladar Fleischman Faculty of Engineering
Tel Aviv University
Tel Aviv, Israel

1 Additions to section 4: Proposed Method

1.1 Ground Truth and Input Generation

As explained in this section, we up-sample the light field grid by adding more samples in-between the given views, while maintaining the original disparity range of the 9x9 light field. This was done using the algorithm in [4]. We train their network to get an input of 2X2 and output a 3X3 grid. Then we run the algorithm on 2X2 segments of the whole 9X9 field, creating a 17x17 field. This can be seen in figure 1. We refer to the 17x17 synthetic light field data as our initial dataset.

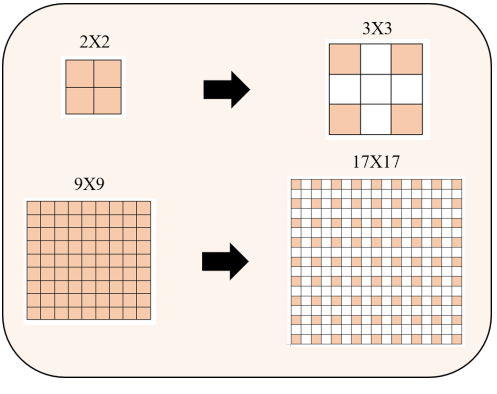

Figure 1: Illustration of creating a denser light field grid by computing in-between views of the given images. We divide the 9x9 grid to blocks of 2x2 and feed them to a synthesizing net [4], which computes a 3x3 block from each of them. Then we arrange the blocks together to create a 17x17 rectangular grid.

In addition, we only use part of the 17X17 input views, which are arranged in a circular shape. Thus, appearing as close as can be to the shape of the camera aperture. In total, we used 241 views out of the 289 views (see figure 2). Regarding the network's input, we choose four views which are arranged in the shape of a rhombus. As described in the main

paper, this was done to match the pin-holes of a light field camera demo developed in our lab.

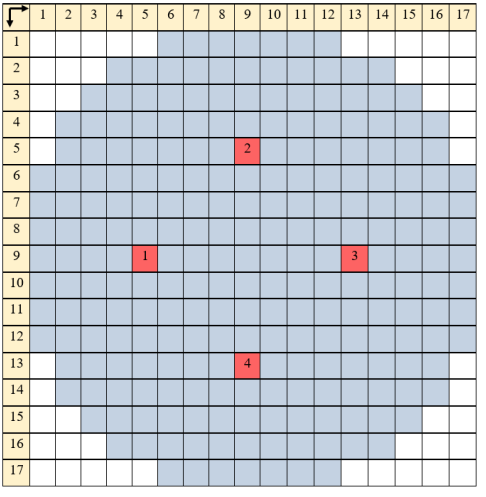

Figure 2: Illustration of the 241 views that are used to compute the ground truth image, arranged in the form of a circle to appear close to a circular aperture in the optical device. The 4 views that are used as an input to the network are marked in red and numbered according to their order in the input tensor.

2 Additions to section 5: Experiments

2.1 Network Architectures

As we are not aware of any previous work that suggests a CNN that performs light field based refocusing, we compare our model to three popular CNN architectures. The networks we evaluate are described below, and their performance on the test set is summarized in tables 1 and 2.

1. Simple CNN: A simple CNN with 7 layers, with constant filter sizes of [3,3] and 128 channels. Each layer is followed by ReLU and batch-normalization [3].

2. ResNet: A modified implementation of ResNet [1] with four residual blocks, each block consists of two layers. In addition, there is a convolutional layer before the residual blocks and a convolutional layer after the residual blocks. We use ReLU and batch-normalization within each layer, except in the last layer. All the layers have constant filter sizes of [3,3] and 64 channels.

3. DenseNet: A modified implementation of DenseNet [2], with two residual blocks, each block consists of two sub-blocks. Each sub-clock consists of two convolutional layers, one with a filter size of [1,1] and the other one with a filter size of [3,3]. The first sub-block has 6 channels in each layer, and since the outputs are concatenated the output has 12 channels. The second sub-block has 12 channels in each layer, resulting in an output of 24 channels. After each dense block, there is a transition layer, reducing the number of channels from 24 to 6. Each layer is followed by ReLU and batch-normalization.

The loss function used to train all architectures is:

$$\ell_{refocus} = \sum_{j=1}^{B} \|\hat{P}_j - P_j\|_1, \tag{1}$$

where B is the batch size, $\hat{P}_j$ is the refocused reconstruction and P_j is the ground truth patch. We choose the ℓ_1 loss as it is more robust to outliers in the training data compared to ℓ_2.

Clean Data - Input views: 2x2			
	PSNR	**SSIM**	**Time (Sec)**
RefocusNet	**46.15**	**0.997**	0.023
ResNet[1]	41.87	0.994	0.031
Simple CNN	40.57	0.994	0.043
DenseNet [2]	37.09	0.982	0.013

Table 1: Results of light field refocusing with 2x2 given input views using four different CNN architectures.

Noisy Data - Input views: 2x2			
	PSNR	**SSIM**	**Time (Sec)**
RefocusNet	**40.95**	**0.990**	0.022
ResNet[1]	38.87	0.987	0.031
Simple CNN	38.90	0.988	0.044
DenseNet [2]	33.88	0.970	0.013

Table 2: Results of light field refocusing on noisy data with 2x2 given input views using four different CNN architectures.

2.2 Computational efficiency

An important advantage of our method is that it has low memory usage, and therefore can be used for applications on mobile devices, where the storage is very limited. Table 3 summarizes the storage properties of each of our architectures and Kalantari *et al.* [9], when the later is divided into two ways of computation during test time: parallel computation and sequential computation. The parameter **network size** includes the sizes of the weights and biases of the networks. The parameter **memory** includes the size of occupied memory required for the activations in inference time. For Kalantari *et al.* , during **parallel computation** the memory is multiplied by 17^2 while during **sequential computation** the computational time is multiplied by 17^2, where 17^2 is the number of light field images synthesized in the process.

As can be seen in Table 3, our most efficient network, RefocusNet, requires about 760 times less memory than Kalantari *et al.* in the parallel case and about 2.6 times less memory in the sequential case; however, in the latter case their algorithm requires 7 orders of magnitude of computational time more than ours.

Architecture	Net. Size (MB)	Memory (MB)	Time at Test (Secs.)
RefocusNet	0.94	50.16	0.023
Simple CNN	3.02	84.48	0.043
ResNet [1]	1.22	84.48	0.031
DenseNet[2]	0.02	15.84	0.013
Kalantari *et al.* [4]- Parallel	6.58	38,148	3,198
Kalantari *et al.* [4]- Sequential	6.58	132	924,222

Table 3: Comparison of different network architectures and Kalantari *et al.* [4] in terms of memory used (including activations) and computational time for an image of size 332x497x3.

2.3 Refocusing Ablation Study

2.3.1 Number of Inputs

We turn to explore the influence of the number of sub-aperture views given as an input to the neural network on the refocusing quality. We train it on (i) 2 input views, (ii) 4 input views shaped like a rhombus, (iii) 4 input views shaped like a rectangle and (iv) 8 input views shaped as the union of the two combinations of the 4 views. We also compute the refocused image using the traditional shifting and averaging method [5, 6, 9] as a reference.

2 Horizontal Views		
Architectures	**PSNR/SSIM**	**Time (Sec)**
RefocusNet	42.17 / 0.994	0.021
Traditional [5, 6, 9]	31.63 / 0.884	14.58
4 Rectangular Views		
Architectures	**PSNR/SSIM**	**Time (Sec)**
RefocusNet	44.74 / 0.996	0.023
Traditional [5, 6, 9]	37.82 / 0.948	15.94
4 Rhomboid Views		
Architectures	**PSNR/SSIM**	**Time (Sec)**
RefocusNet	46.15 / 0.997	0.023
Traditional [5, 6, 9]	35.34 / 0.941	16.11
8 Views		
Architectures	**PSNR/SSIM**	**Time (Sec)**
RefocusNet	48.21 / 0.998	0.026
Traditional [5, 6, 9]	39.40 / 0.971	23.70

Table 4: Sub-aperture views ablation study. Comparing between different input sizes: 2 sub-aperture views, 4 sub-aperture views shaped as a rectangle, 4 sub-aperture views shaped as a rhombus, and 8 sub-aperture views. The results are averaged over 20 test images.

As can be seen in table 4, the highest improvements in the PSNR values of RefocusNet compared to the traditional approach occur in the use-cases of 2 horizontal views and 4 rhomboid views (around 10 dB). Our network's performance increases constantly when increasing the number of sub-aperture input views, and when changing the shape of 4 input views from rectangle to rhombus. The computational time at test time increases with the

number of sub-aperture views but in a negligible range of a few hundredths of a second. We can also see that, as expected, when applying the traditional refocusing method of shifting and averaging, increasing the number of sub-aperture views results in a consistent increase in the PSNR values as well as the computational time (due to the added information). We may conclude that for a refocusing neural network, the optimal number of sub-aperture views needed as input is 4 rhomboid views; since in these ranges of PSNR (45 dB and higher), the improvements are barely seen to the human eye. Using 8 input views will require higher angular resolution of the capturing device and higher computational complexity. Yet, all of this may lead to a marginal benefit.

2.3.2 Number of Layers

We also tested the influence of the network's depth on its performance. Table 5 shows that increasing the number of layers from 7 to 9 and 11 only improves the performance by marginal values and increases the computational time. Considering the trade-off between performance and computational time and the principal of vanishing returns, we decided to choose an architecture with 7 layers.

Number of layers	PSNR/SSIM	Time [sec]
3	43.44/0.995	0.012
5	45.39/0.996	0.017
7	46.15/0.997	0.023
9	46.53/0.997	0.029
11	46.23/0.997	0.035

Table 5: Performance of RefocusNet as a function of the network's depth.

2.3.3 Contribution of Skip Connections

In order to estimate the contribution of the skip connections in RefocusNet, we trained an almost identical network - but without the skip connections. The performance during inference decreased significantly - more than 10 dB (from 46.21 to 35.33), which shows us that adding the residual layers is crucial to the network performance. The computational time, however, also decreased to 0.017 seconds. A visualization of the residual layers can be seen in figure 3.

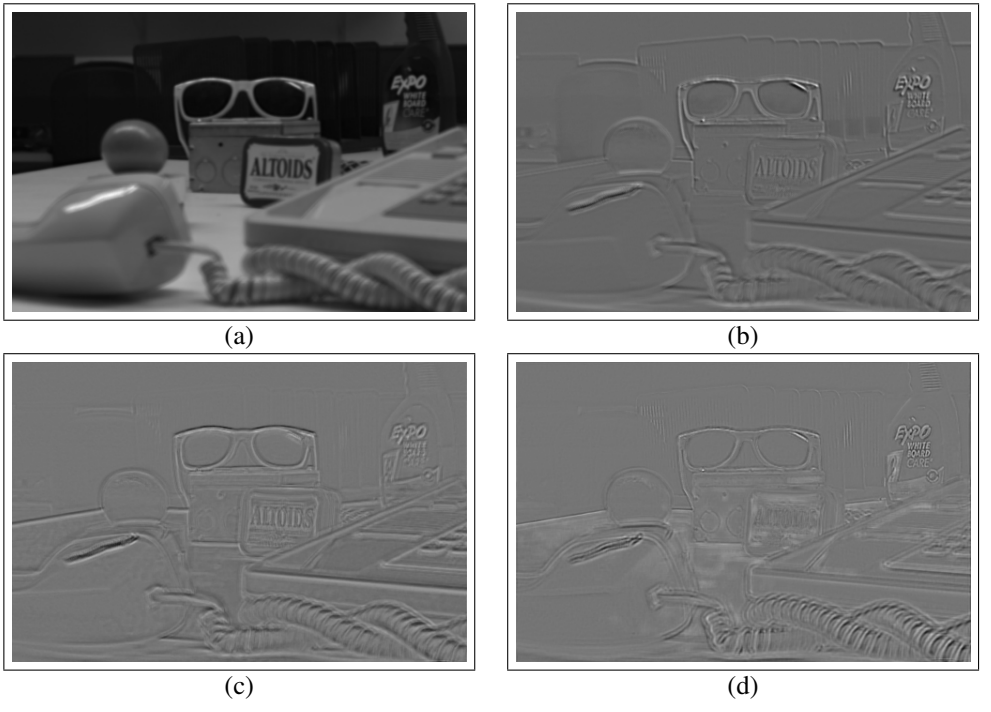

Figure 3: Visualization of the 1st, 3rd, 5th and 7th residual layers. (a) residual layer 1; (b) residual layer 3; (c) residual layer 5; (d) residual layer 7.

2.3.4 RefocusNet Variations

As mentioned in section 4 in the main paper, part of our RefocusNet was based on an existing architecture named DenoiseNet ([7, 8]). We will now explain the motivations that led us to build the final architecture. At first, we only added a second input to the architecture: the average of the four input views, which makes the network robust to different focus sizes. A diagram of this architecture is presented in figure 4. While the PSNR results on clean data were great - 46.21 dB, the results on noisy data were not satisfying - 39.95 dB. We figured that the noise appearing in the second input is the cause.

Next, we decided to take this averaging from the sum of residuals, figuring that at this point, the network probably cleaned the noise. A diagram of this architecture is presented in figure 5. Although the PSNR results on noisy data improved to 40.17 dB, the results on clean data degraded to 45.85 dB.

Finally, we decided to keep using the average calculate from the sum of residuals, but also to add an additional convolution layer on the sum of residuals and then sum these both layers together. The PSNR results on clean data improved back to 46.15 dB, and results on noisy data also improved to 40.96 dB. This is the final architecture shown in the paper.

This study shows the contribution of this specific architecture to our problem: it preserves the parts in the refocused image that we wish to preserve (the parts where the object is in focus), and repairs the parts that need smoothing e.g. the parts with the ghosting artifacts.

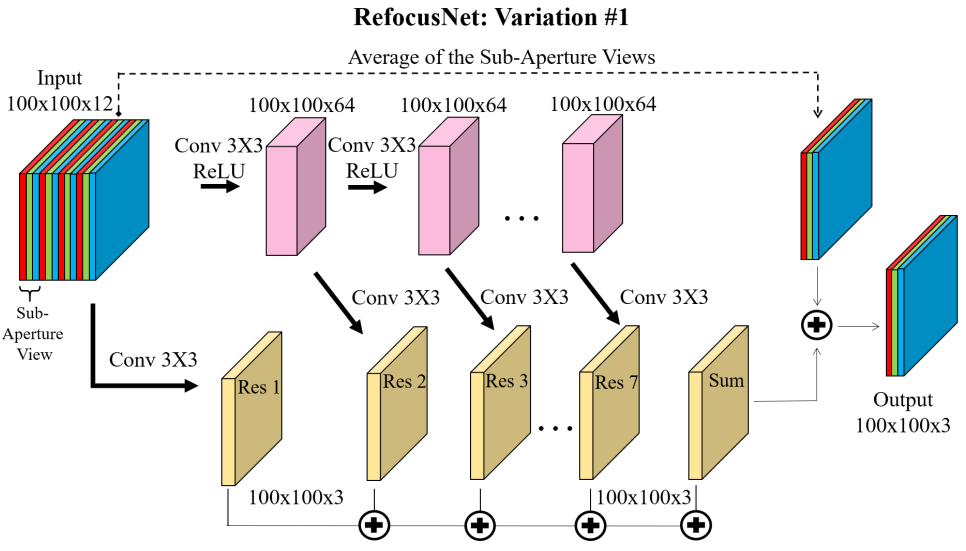

Figure 4: RefocusNet variation 1. Performance [PSNR/SSIM]: clean data - 46.21/0.997, noisy data - 39.95/0.989

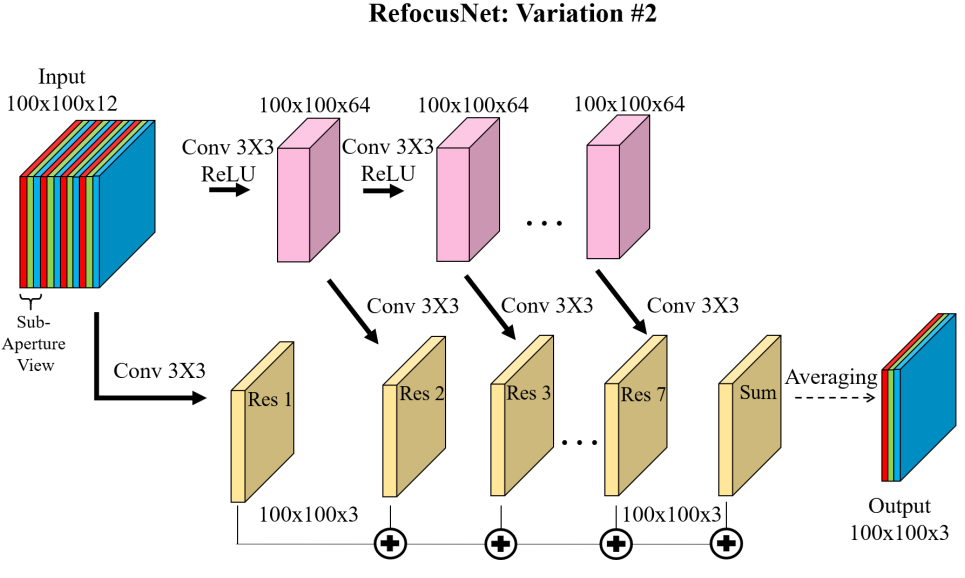

Figure 5: RefocusNet variation 2. Performance [PSNR/SSIM]: clean data - 45.85/0.997, noisy data - 40.17/0.990

2.3.5 Noisy Data

We performed an ablation study on noisy data, using Gaussian noise with increased standard deviations. The network was trained using noise augmentation, with noise standard deviation distributed uniformly at random from the range [0, 0.08]. During test time, we used standard deviation ranges from [0, 0.2]. Table 6 shows the results of RefocusNet and the results of Kalantari *et al.* [4], divided into two sections: results on noisy data with standard deviations that the network was trained on ([0,0.08]) and results on standard deviations that were not seen during training ([0.1, 0.2]).

As expected, the performance degrades when increasing the noise's standard deviation, but our network still performs well on noises that were introduced to it during inference for the first time. Also, our method outperforms Kalantari *et al.* [4] at all standard deviations except for the largest (0.2), and the difference is decreasing when increasing the standard deviations (as it becomes farther apart from the range seen during training).

It is important to note that the original algorithm of Kalantari *et al.* [4] does not pad the convolutional layers, and also crops the results to avoid border effects created by the receptive fields during training. Thus, they crop the output images by 22 pixels from each side in both dimensions. Yet, since the network only synthesizes the angle views and does not refocus them, the refocused result images still have border effects as mentioned in Section 5 in the paper (6 pixels from each side). In order not to lose that much information from the images, we re-trained their network with padding in each convolutional layer and canceled the manual cropping at the end of the synthesizing. We only cropped 6 pixels from each side after refocusing, as we did with our networks. This operation actually improved the performance of their algorithm (PSNR improvement of $\sim$2dB), both on clean and noisy data, but not beyond RefocusNet.

Standard Deviations in the range [0.02,0.08]		
σ	**Ours PSNR/SSIM**	**Kalantari *et al.* [4] PSNR/SSIM**
0.02	44.44/0.996	42.13/0.988
0.04	43.23/0.995	41.36/0.985
0.06	42.05/0.992	40.45/0.980
0.08	40.95/0.991	39.45/0.974
Standard Deviations outside the range [0.02,0.08]		
σ	**PSNR/SSIM**	**Kalantari *et al.* [4] PSNR/SSIM**
0.1	39.93/0.988	38.48/0.966
0.12	38.91/0.985	37.51/0.956
0.14	37.81/0.981	36.54/0.945
0.16	36.50/0.976	35.67/0.933
0.18	34.90/0.969	34.85/0.918
0.2	32.97/0.985	34.11/0.904

Table 6: Performance of RefocusNet and Kalantari *et al.* [4] on noisy data, divided into two sections: results on noisy data with standard deviations that the networks were trained on between the range of [0.02,0.08], and results on standard deviations that the networks were not trained on.

3 Additional Refocusing Results

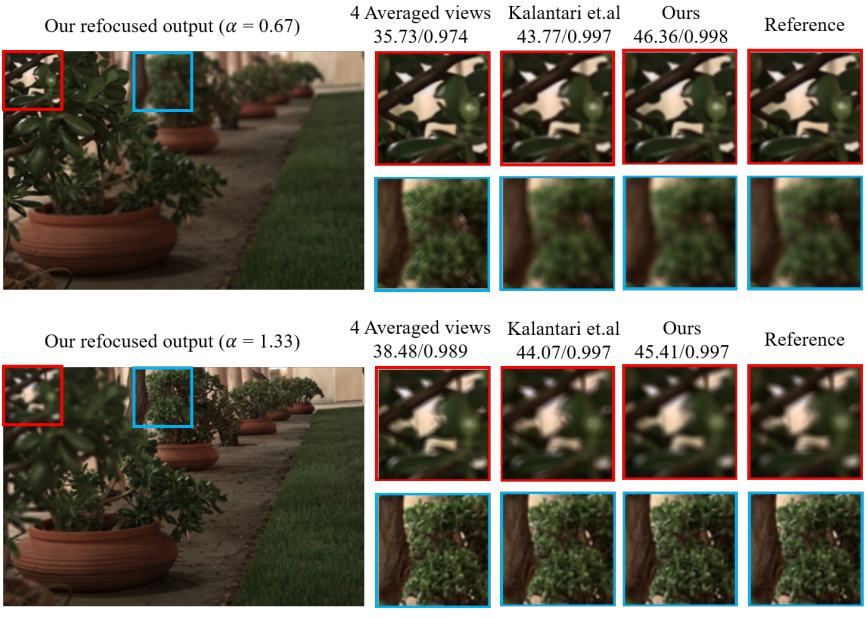

Figure 6: Comparison of refocusing results (PSNR/SSIM) between the traditional method of shifting and averaging with 4 given sub-aperture views ([5, 6, 9]), Kalantari *et al.* [4] and our RefocusNet. Our method generates a continuous refocused image, which is very similar to the reference image (calculated using 241 sub-aperture views). However, the results of Kalantari *et al.* still have some ghosting artifacts around the leaves (see the red patch of focus 1.33).

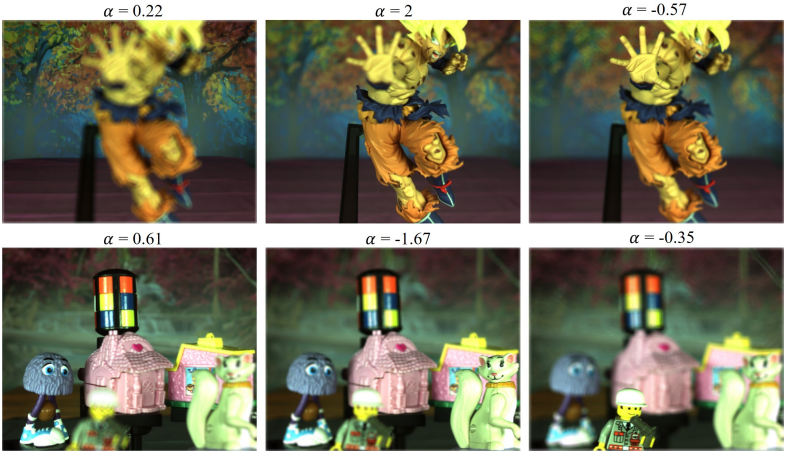

Figure 7: Additional results of refocusing on images taken with our demo camera using RefocusNet. Note the exact details of the Lego-man in the bottom right image.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[4] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6): 193, 2016.

[5] Marc Levoy, Billy Chen, Vaibhav Vaish, Mark Horowitz, Ian McDowall, and Mark Bolas. Synthetic aperture confocal imaging. *ACM Transactions on Graphics (ToG)*, 23(3): 825–834, 2004.

[6] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, Pat Hanrahan, et al. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11):1–11, 2005.

[7] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Deep class-aware image denoising. In *IEEE International Conference on Image Processing (ICIP)*, pages 1895–1899, Sep. 2017.

[8] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Class-aware fully convolutional gaussian and poisson denoising. *IEEE Transactions on Image Processing*, 27(11):5707–5722, Nov 2018.

[9] Vaibhav Vaish, Bennett Wilburn, Neel Joshi, and Marc Levoy. Using plane+ parallax for calibrating dense camera arrays. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.