Link to project source code: https://github.com/varunshenoy/UDC-Image-Restoration

A Deep Learning Approach for Image Reconstruction from Smartphone Under Display Camera Technology

Arjun Dhawan and Varun Shenoy

Abstract—In this paper, we present a method for reconstructing images taken from a under display camera. Due to the placement of an under display camera, resulting images suffered from optical aberrations and quality degradation. The method we propose utilizes a convolutional neural network to generate a high quality image from a low quality, distorted input image. A under display camera dataset containing Transparent OLED (T-OLED) images is used to train the neural network and a portion of the data is used to evaluate the models performance. Our propsed shallow ResNet method generates reconstructed images that are quantitatively and qualitatively better than the inputs. Quantitatively, the model increases the PSNR of the input images by 2.39 on average. Qualitatively, the model succeeds at sharpening and deblurring the images. However, the model struggles with preserving color, correcting in low-light environments, and eliminating vertical fringe artifacts in the image. We propose methods to fix these issues and generate more visually appealing images.

1 Introduction

Under display camera (UDC) imaging has been a problem of interest for several years in industry as smartphones move to have full-frame displays without visible camera or sensor peripheries on the front. A diagram of a UDC setup is shown in Fig. 1. Several smartphones with under-display cameras already exist, particularly ZTE's Axon 20 and the Galaxy Z Fold 3 [1]. However, the images resulting from a UDC are of much poorer quality. Companies leverage computer vision algorithms to clean up these pictures, leading to over-processed and unnatural final images. Therefore, accurate algorithms with fast inference are necessary to make under display cameras more ubiquitious.

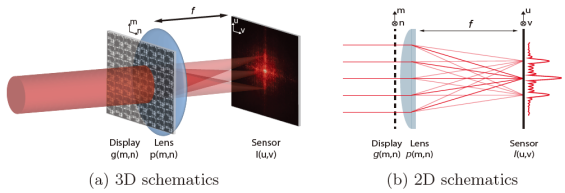
Modern computing environments have allowed for a wide array of new deep learning techniques that were unfeasible before.

A type of deep learning neural network, called a convolutional neural network (CNN), has demonstrated capabilities for highly accurate image reconstruction and mapping after being trained on a large dataset of samples. In the past decade, research efforts have led to impressive results on image segmentation [2], deblurring [3], and denoising [4].

In this paper, we propose an approach to improving the quality of UDC images through a CNN. We introduce a simple deep learning architecture composed of nine successive residual blocks, which contain series of convolutions and nonlinearities. In the end, this approach proved fruitful, improving image quality by a signficant amount. Moreover, the inference time of the residual network is much faster than comparable approaches.

2 Related Work

Resolving UDC images is directly related to the more general approach of deblurring images. When the blur kernel is known, well known signal processing approaches, such as Wiener filtering, or iterative methods, such as HQS [6] or ADMM [7], can be leveraged. However, when the mechanics of the blurring are unknown, this turns into a blind deconvolution problem. While more classical approaches such as Richardson-Lucy [8] have been recommended in the past for blind deconvolution problems, advances in compute hardware and algorithms have enabled this type of problem to be conducive to deep learning techniques.



(a) 3D schematics (b) 2D schematics

Fig. 1. A 3D and 2D schematic for the optical setup for an under display camera from the CVPR 2021 "Image Restoration for Under-Display Camera" challenge [5]

The first massive research driven collaboration towards solving this problem was undertaken at CVPR 2021 under the "Image Restoration for Under-Display Camera" challenge. Several teams from different university and industry research settings competed, most of whom used a combination of classical signal processing techniques and deep learning [5].

The challenge provided a database of UDC images for two types of displays: a 4k Transparent OLED (T-OLED) and a Pentile OLED (P-OLED).

The dataset contains 240 pairs of 1024×2048 images. Challenge participants were evaluated primarily on PSNR. Example images from the dataset are shown in Fig. 2.

Visually, the T-OLED reconstruction task involves a combination of deblurring and the removal of vertical fringes. Fig. 3 displays the blur and vertical fringes of a T-OLED

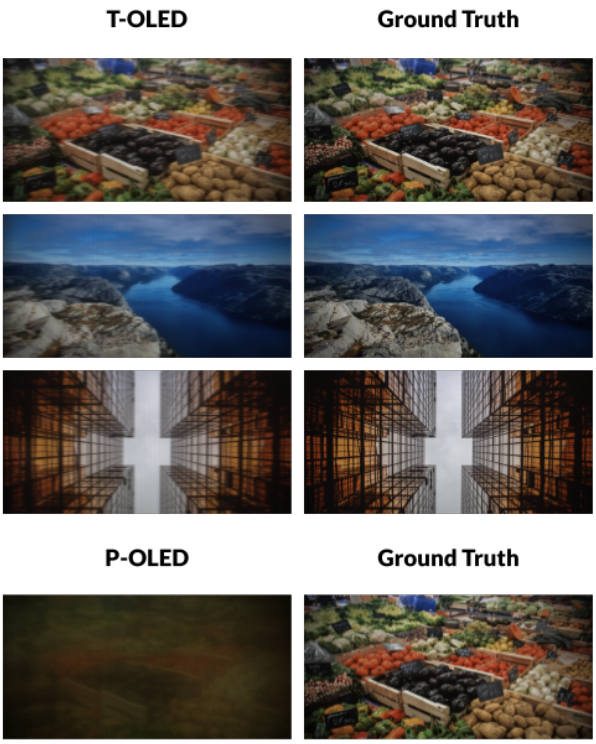


Fig. 2. Examples of images from the dataset. Reconstructing P-OLED images is a much more challenging task, although the same methods may be useful.

image in comparison to its ground truth in closer detail. The P-OLED task is much more challenging, as it involves severe image degradation and desaturation in certain cases. As such, this paper presents an approach for the T-OLED task as there was a limited amount of time allocated for the completion of this project. The approach presented in this paper will be compared to various entries to this competition in a later section.

Convolutional neural networks have been studied and tuned for various image to image mapping tasks. A common network architecture is the Residual Netowrk, or ResNet. ResNet architectures are composed of ResNet blocks, each of which apply a convolution and some kind of batch normaliztion with an activation function at the end. Carefully tuned ResNet architectures have been shown to succeed on image processing tasks like denoising [9].

3 Proposed Method

After a brief experimentation period of using signal processing techniques, such as lowpass filtering to remove the vertical fringes and pretrained denoising models, it was clear that a new approach would be necessary.

Convolutional neural networks vastly outperform other current machine learning models for large scale image processing and reconstruction. Due to computational constraint but general interest in the problem space, our proposed method involves training a shallow ResNet from scratch.

Each ResNet Block contains a 2D convolution layer (with padding to ensure the image is not compressed), batch normalization, and a leaky ReLU activation function. Our

Fig. 3. A larger single example to highlight the differences between the T-OLED display and ground truth. Notice the vertical fringing and blurring that clearly occurs on the T-OLED side of this hybrid image.

entire model is composed of 9 ResNet Blocks with an additional convolution layer at the front and end. A graphical representation of our model is shown in Fig. 4.

3.1 Earlier Approaches

This was our third approach we tried after two simpler ones. We initially assumed that there might be a simple 2D filter (modeled by a spatial convolution) that could be uncovered from the input and output pair for a given image. The first test involved using a basic least squares approach on $Ax = b$, where A was the 2D convolutional matrix for the input image, x was the convoutional kernel, and b was the vectorized output image.

Our second test set up a model with a single 2D convolutional layer, and we tried training it on the dataset. We soon realized the infeasibility of both these approaches due to its naivete. There was no clear linear mapping that could clean up the image simply from each pixel's local neighborhood (seen from the convolutional window).

3.2 Tools

We used Pytorch to train our models on a Tesla K80 GPU in the Google Colab environment.

3.3 Data Preprocessing

We partition the dataset into training and validation sets. 80% of the data is used as the training data for our model, 10% is used for validation, and 10% is left for model evaluation and testing. The validation dataset is used for hyperparameter turning.

To fit our data in the cloud environment, we needed to also scale the dataset to 650×325. This presents a more challenging task than using the normal sized images since we have lost a significant amount of information.

Due to the limited amount of data available, we also experimented with data augmentation, randomly flipping images horizontally or vertically during the training process.

3.4 Model Training

After generating our 9-layer ResNet model, we trained it while turning several hyperparameters. We chose to use the

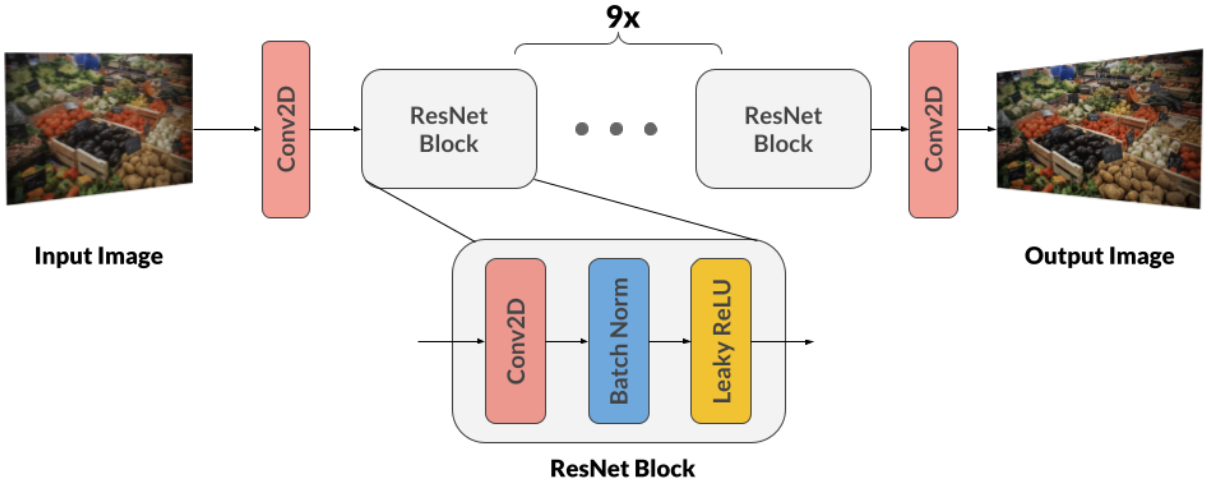


Fig. 4. A high level overview of our model's architecture.

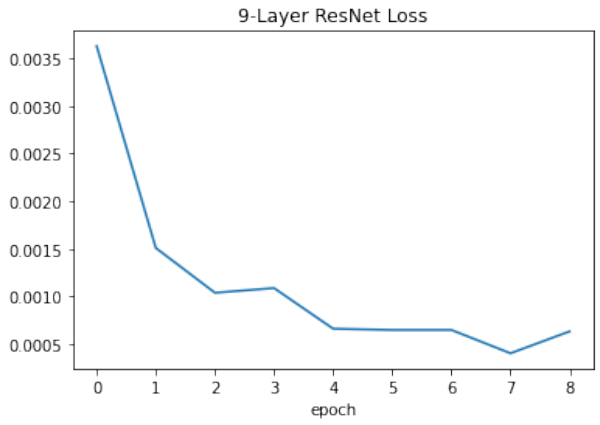


Fig. 5. Loss function while training model

$\ell 1$ loss function as prior research in image reconstruction has suggested $\ell 1$ loss allows for sharper image formation (compared to, for example, mean squared error) [10].

We used the Adam optimizer with a learning rate of 0.01. Our final model was trained for 8 epochs, attaining a final running loss of 6.3×10^{-4} starting with a running loss of $3.63 \times 10^{-}3$ from the first epoch. Fig. 5 shows that our model achieved convergence after 8 epochs. Even when training our model for more epochs, we found that the loss did not change much from there.

Our weights were all initialized to zero. In the future, we hope to test out different intialization methods and optimizers to see if any improvements upon this baseline can be made.

4 Comparison to Other Methods

We compare our results and methods to three approaches in the UDC challenge at CVPR as they were constrained to the same dataset as us. Our leading metric for comparison is the Peak Signal-to-Noise Ratio (PSNR), which can be described as

$$PSNR = 10 \log_{10} \left(\frac{\max(I)}{MSE} \right)$$

where

$$MSE = \frac{1}{N} \Sigma_{n=1}^{N} (I_n - \hat{I}_n)^2.$$

The leading team for the challenge, Baidu Research Vision, used a convolutional neural network composed of a series of dense residual blocks alongside a shade-correction preprocessing step. The shade-correction mechanism multiplicatively applies a certain pattern to a given patch of a T-OLED display before sending it off to the neural network. The network was trained on patches of size 128×128. This team achieved an average PSNR of 38.23 on the test set along with an inference time of 11.8 s/frame on a Tesla M40.

Team BigGuy applied a much simpler approach. They trained a large modified U-Net with patches from the images. They also tested several different types of residual dense blocks and neural architectures, arriving at the conclusion that the U-Net architecture would obtain the best results. This team scored a PSNR of 38.3 with an inference time of 0.3548 on a Tesla V100.

Team San Jose Earthquakes used two stages for solving the T-OLED problem. The first stage passes the input to a large network PyNET in order to deblur and colorize the image. They also pass the input through an ADMM module to obtain a poorer, noisier reconstruction. Both the ADMM and PyNET outputs are then passed into a fusion U-Net to recover a sharper, denoised result. The authors remark that the ADMM output maintained high frequency textures while the PyNET output often blurred them out, and thus rose to the conclusion of passing both through a fusion U-Net. However, due to the computational complexity of ADMM on large images, they obtained an inference time of 180 s/frame with a PSNR of 33.78.

In comparison, our methodology is most similar to that of Team BigGuy with added simplicity. Rather than using patches, we simply train our model with a scaled down version of the image. The patch based approach makes sense when training U-Net since it requires fixed-size, square input images. While Baidu Research Vision and San Jose Earthquakes were much more detailed in their approaches, their algorithms would be impractical from an inference

point of view, especially considering that these models would need to run on a smartphone. Our model had an inference time of approximately 1 s/frame on an Apple M1 chip, which would be even faster on similar GPUs to those used by these other teams. Thus, the simple model presented in this paper has its merits.

5 Results

In order to evaluate and test our model, we performed inference on the remaining 10% of the dataset. We kept track

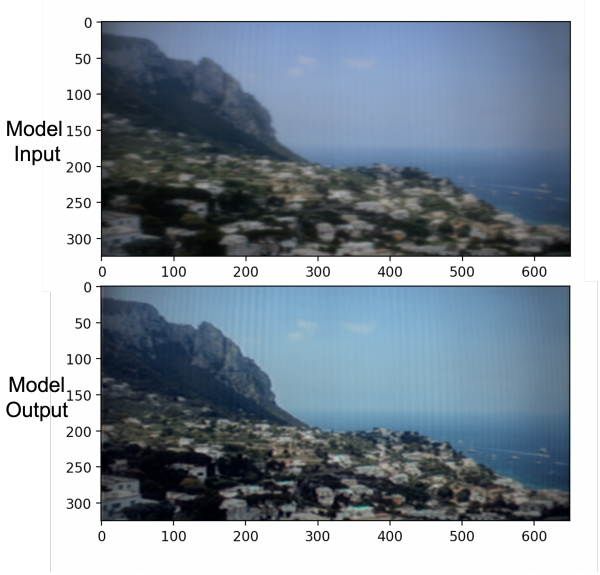


Fig. 6. Average example of input/output of model

of the input and output of the model as well as the high res equivalent stored within the dataset.

Using these the input, ouput, and ground truth image, we were able calculate PSNRs of the input image compared to the high resolution reference image, the output image compared to the high resolution reference image, and the difference between those PSNRs. Two examples from the test set are shown in Fig. 7. We document the best and worst PSNRs seen in the test dataset to give an idea of the range of outputs expected.Notice how the image our model fails on is possibly the most challenging in the entire dataset. It is in extremely low light, and it is hard to discern basic features, even in the high resolution reference image. On the other hand, it is clear that our model was successful at deblurring the other example. However, it fails to maintain color accuracy. At a closer look, we can also recognize the fringing pattern from the original input image. These issues can likely be resolved with a more robust training approach (larger network, more data augmentation, more training time) that would be not difficult to implement with additional compute power.

On average, the PSNRs for the model output compared to the high resolution reference was 33.09 whereas the PSNRs for the model input compared to the high resolution reference was 30.7. There is a definitive improvement in the PSNR values as the model output PSNR is larger than the model input PSNR. While PSNR does not completely characterize the human visual response to an image, this metric indicates that the model is producing a positive output. Additionally, as Fig. 6 is representative of the average input and output from the model, there is a definitive visual improvement in to output. Qualitatively, the output tends to be sharper, brighter, and less hazy than the input images. However, the vertical artifacts seem to have been amplified by the deblurring mechanism. Patch restoration approaches, such as those described in prior work, appears to be a good solution for eliminating these stripes. One other possible approach is using low-pass filtering to eliminate the bands since they all seem to be periodic and predictable.

From a quantitative perspective, the PSNRs of the model inputs and model outputs are plotted to understand the distribution across our test dataset. In Fig. 8 , we can see a histogram of this distribution. Since the data is not normally distributed, it is difficult to understand the spread of the data with normal metrics such as standard deviation.

Instead we look at the mean of the original and reconstructed PSNRs as well as their mean absolute deviations (MAD). The original PSNRs (model inputs) has a mean of 30.7 with a mean absolute deviation of 1.9. The reconstructed PSNRs (model outputs) has a mean of 33.09 with a mean absolute deviation of 1.7. The means differ significantly while the mean absolute deviations are relatively similar between the inputs and outputs . This indicates that the model effectively shifted up the PSNRs of the test dataset. Quantitatively, the model has increased the mean PSNR value of the dataset while retaining a similar distribution of PSNRs.

As our final metric of improvement with this model, we look at the empirical difference in PSNR values between the input and outputs. Fig. 9 shows the histogram of differences in PSNR values.

The plot of differences is of the reconstructed PSNR values minus the original PSNRS. We can see that on average there is an improvement in the output images. The outliers are evident as there are some points in the dataset with a negative PSNR difference indicating a quantative worse output than input. The worst and best case PSNR values are displayed in Fig. 7. While the PSNR difference is negative for our worst output, qualitatively, the image still seems to be sharper, brighter, and closer to the high resolution reference. This is indicative of the PSNR metric not fully characterizing the human visual response.

6 Discussion

In understanding the results section above, it is imperative to note that the output of the model cannot be evaluated by quantative metrics only. It is also important to evaluate the images qualitatively.

Upon a visual inspection of our models outputs, it was evident that it was successful at deblurring the image and outputting a sharper and more well defined image. However, we note that we had poor color color correction. Based on our literature review, it is evident that our method is similar to Team BigGuy's method. Our approach used a smaller network and did not use nearly as many GPU's. We hypothesize that by using more data (more permutations and augmentations of the training data) as well as increasing the size of our network, we could improve the color correctness output of the model.

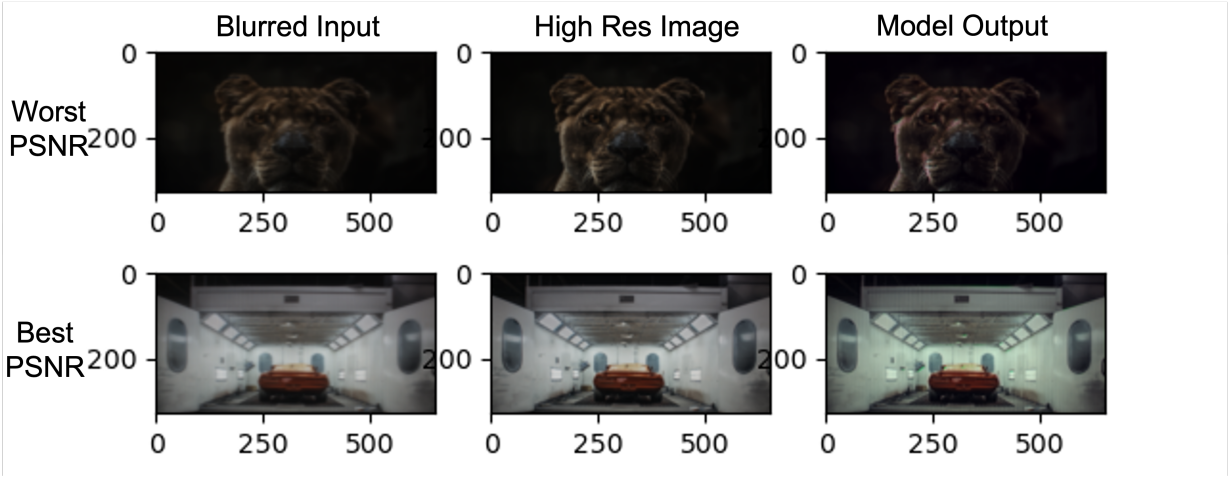


Fig. 7. Input, Output, and Reference images for trained model.

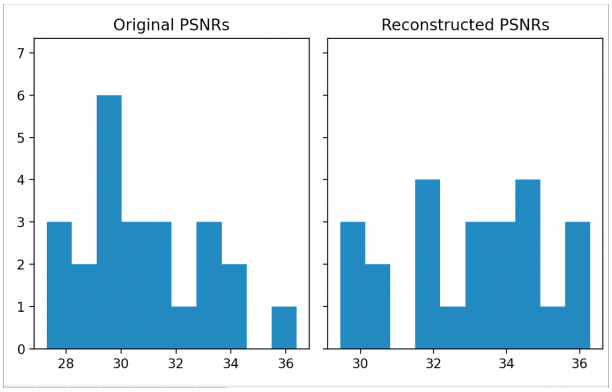


Fig. 8. Histogram of PSNRs for model input/output.

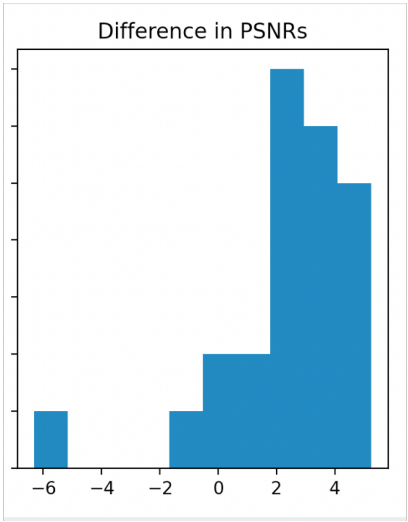


Fig. 9. Histogram of difference in PSNRs.

Additionally, the model fails to remove the fringes seen in the input images resulting from the pixels in the display. In future iterations of this project, this could be fixed with classical digital signal processing techniques or even image inpainting after lowpass filtering the image to get rid of the fringes. We could also adopt patch based methods shown in prior work, as the fringes appear to be predictable.

From a quantitative perspective, it is notable that different lighting situations as well as high frequency portion of input images proves to be difficult for the model. Qualitatively images in these conditions seem to have slightly sharper outputs, however, the worst PSNR output values compared to PSNR input values are a result low light and high frequency images. One possible solution to this is adding a normalization step in preprocessing. This could ensure the input has an expected amount of brightness that would not need to be learned by the convolutional neural network.

In an commercial setting where algorithms/models are developed for specific cameras and displays, we will have fully characterized optical models for the cameras and displays being used. Therefore, we can account for the specific points spread functions and design the algorithms around the hardware. We could use DeepOptics [11] to design our hardware for our specific algorithms and improve image outputs.

In conclusion, our proposed method definitely produces a better quality image than the input and is a step in the right direction. With a better understanding of the camera and display technology used as well as larger networks and more GPUs to train our model on, we believe our method could utilize under display camera technology could output high quality images.

7 Source Code

Our source code and details on how to get our implementation working are available on GitHub at https://github.com/varunshenoy/UDC-Image-Restoration.

References

[1] S. Byford, "Under-display cameras are slowly getting better," Nov 2021. [Online]. Available: https://www.theverge.com/22776271/under-displ

[2] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[3] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin, "Deep learning on image denoising: An overview," *Neural Networks*, vol. 131, pp. 251–275, 2020.

[5] Y. Zhou, M. Kwan, K. Tolentino, N. Emerton, S. Lim, T. Large, L. Fu, Z. Pan, B. Li, Q. Yang, Y. Liu, J. Tang, T. Ku, S. Ma, B. Hu, J. Wang, D. Puthussery, P. S. Hrishikesh, M. Kuriakose, C. V. Jiji, V. Sundar, S. Hegde, D. Kothandaraman, K. Mitra, A. Jassal, N. A. Shah, S. Nathan, N. A. E. Rahel, D. Chen, S. Nie, S. Yin, C. Ma, H. Wang, T. Zhao, S. Zhao, J. Rego, H. Chen, S. Li, Z. Hu, K. W. Lau, L.-M. Po, D. Yu, Y. A. U. Rehman, Y. Li, and L. Xing, "Udc 2020 challenge on image restoration of under-display camera: Methods and results," in *Computer Vision – ECCV 2020 Workshops*, A. Bartoli and A. Fusiello, Eds. Cham: Springer International Publishing, 2020, pp. 337–351.

[6] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE transactions on Image Processing*, vol. 4, no. 7, pp. 932–946, 1995.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[8] D. Fish, A. Brinicombe, E. Pike, and J. Walker, "Blind deconvolution by means of the richardson–lucy algorithm," *JOSA A*, vol. 12, no. 1, pp. 58–65, 1995.

[9] H. Ren, M. El-Khamy, and J. Lee, "Dn-resnet: Efficient deep residual network for image denoising," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 215–230.

[10] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on computational imaging*, vol. 3, no. 1, pp. 47–57, 2016.

[11] V. Sitzmann, S. Diamond, Y. Peng, X. Dun, S. Boyd, W. Heidrich, F. Heide, and G. Wetzstein, "End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 114, 2018.