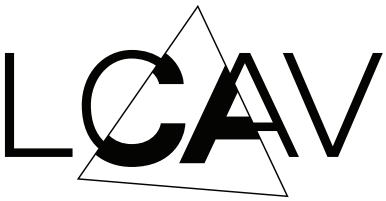

Lensless PiCam

Learning-based technique for lensless reconstruction

Yohann Perron, yohann.perron@polytechnique.org

September 8, 2023

Abstract

In this internship, I explore different optimization algorithms for lensless imaging. Lensless imaging is a new imaging technique that replaces the lens of a camera with a diffuser mask. This allows for simpler and cheaper camera hardware. However, the reconstruction of the image from the sensor measurement is a complex ill-posed problem. The goal of this internship is to improve current reconstruction algorithms for lensless imaging. I first implemented unrolled optimization as a low computational cost way of increasing reconstruction quality. I also explored the possibility of using a plug-and-play algorithm for lensless imaging. Then, I introduced a pre-/post- denoising scheme mixing unrolled optimization and more traditional deep learning which improved current state-of-the-art result on the DiffuserCam Lensless Mirflickr Dataset. Finally, I introduce a framework for learning the mask pattern of the lensless camera jointly with the reconstruction algorithm, allowing for further improvements.

Acknowledgements

First of all, I would like to thank Eric Bezzam, my research supervisor at EPFL. Throughout the internship, he was always present to help and guide me. His numerous pieces of advice, sharing of experience, and overall dedication to the project were immensely helpful.

I would also like to thank my professor at the MVA Master, Jean-Michel Morel, who agreed to be my academic supervisor for this internship. His advice was one of the main inspirations for the pre-/post- denoising scheme presented in chapter 5.

Finally, I would like to extend my sincere appreciation to everyone at the LCAV, especially the Ph.D. students and fellow interns. Working alongside them enabled me to discover new subjects and problems in the field of signal processing and inverse problems. Their warm reception and inclusive nature made me genuinely feel like an integral part of the team and a valued member of the research community.

Table of contents

Chapter 1

Introduction

1.1 History of photography

The origins of photography can be traced back to the concept of the pinhole camera, also known as the camera obscura. Ancient Chinese philosophers and Arab scholars understood the basic principles of light passing through a small hole and projecting an inverted image on a surface opposite to the hole. This principle was further refined by Renaissance artists, who utilized camera obscuras as tools for accurate drawing and perspective. However, it was not until the early 19th century that advancements in materials and optical understanding led to the creation of more practical and portable pinhole cameras.

The mid-19th century marked a pivotal shift with the invention of the daguerreotype process and the introduction of lenses into photography. The integration of lenses improved light gathering, leading to shorter exposure times and enhanced image clarity. This advancement laid the foundation for modern photography and initiated a wave of technological innovations in the field.

As photographic technology continued to evolve, the limitations of lensed cameras prompted researchers and engineers to explore alternative imaging methods. One significant outcome of these explorations was the emergence of lensless imaging techniques. Unlike traditional cameras that rely on lenses to focus light onto a photosensitive surface, lensless imaging techniques utilize diffraction masks to capture and reconstruct images without the need for conventional optics. This approach allows the complexity and cost of hardware to remain low while keeping light-gathering capability similar to traditional lensed cameras. These characteristics are summarized in Table 1.1.

However, lensless imaging forgoes the one-to-one mapping between the scene and sensor that has been used in both pinhole cameras and traditional cameras. This means that the reconstruction of the image from the sensor measurement is a complex ill-posed problem. This problem is usually solved using optimization algorithms with regularization. The goal of this internship is to explore different optimization algorithms for lensless imaging.

1.2 Lensless Camera

A lensless camera is very similar to a traditional lensed camera, except that the lens is replaced by a mask (See figure 1.1). However, this results in a very different image formation process and changes massively the post-processing pipeline. In lensless imaging, a single point in the scene can potentially influence all the pixels on the sensor. This is in contrast with traditional

Aspect	Pinhole Camera	Lensed Camera	Lensless Camera
Light Gathering	Limited light-gathering Longer exposure times Challenging for motion	Improved light-gathering Shorter exposure times Capable of capturing motion	Comparable to lensed
Image Quality	Limited sharpness	Sharper images	Dependent on the reconstruction
Aesthetics	Limitless depth of field Distinctive artistic appeal	Adjustable focus Versatility	Focussing at reconstruction
Complexity	Minimal	Complex optics	Hardware : Minimal Software : High
Cost	Minimal	Higher production cost	Minimal

Table 1.1: Pros and Cons of Pinhole, Traditional Lensed, and Lensless Imaging Cameras

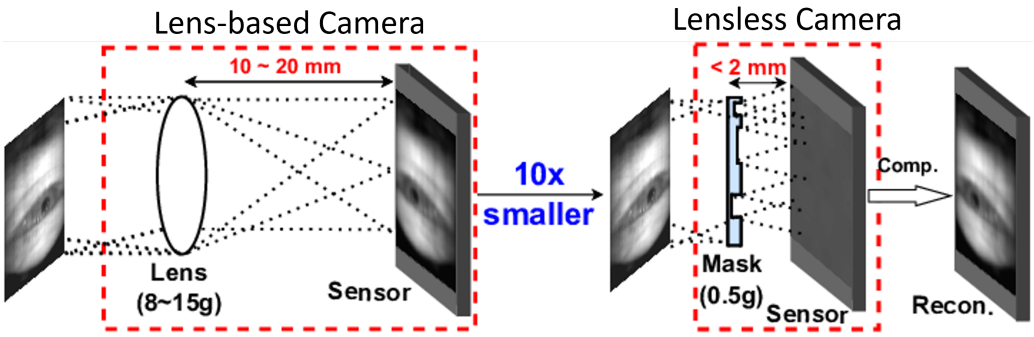

Figure 1.1: Lensless camera vs traditional lensed camera. Original image from [1]

lensed cameras where a single point in the scene can only influence a single pixel on the sensor. A simple but accurate model of the image formation process is given in section 1.3.

1.2.1 Mask

There are two main types of masks used in lensless imaging: amplitude, and phase masks:

An amplitude mask is a fine film mask modulating the intensity of the light passing through it. The pinhole camera can be viewed as a special case of amplitude mask where the mask is a simple hole. However, the more general framework of amplitude mask allows for a greater portion of the light to pass through by using multiple "holes" of different sizes and shapes. This allows for greater light-gathering capabilities and shorter exposure times compared to pinhole cameras while keeping the same simplicity of hardware. Amplitude masks can be easily fabricated by printing the pattern on a sheet of transparent plastic for example. An even more interesting choice is to use the light-modulating portion of an LCD panel to create an inexpensive and reconfigurable amplitude mask.

Phase masks are more complex to fabricate but allow for greater light-gathering capabilities. A phase mask is a mask modulating the phase of the light passing through it, similar to a lens. This means that most of the light passes through the mask onto the sensor. However, manufacturing them is a bit more challenging. A typical approach is to etch a piece of glass as in [2], but it can be challenging to accurately produce a given mask. If one is fine with a random mask, a cheap alternative is to use the natural thickness variation of any transparent material. The camera described in [3], shows success with a simple piece of transparent tape.

The design of the mask can have a great impact on the quality of the reconstructed image. In section 1.3, we describe a general approach able to work with any mask. However, some mask designs allow for specific reconstruction algorithms that can improve the quality or/and speed

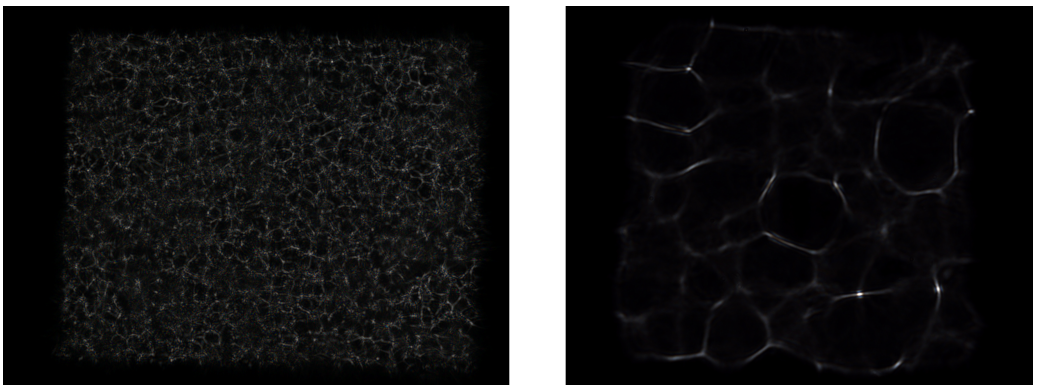

Figure 1.2: On the left: PSF of a simple piece of tape. On the right: PSF of the DiffuserCam Dataset.

of the reconstruction (often with a closed-form solution) [4, 5, 6].

1.2.2 Reconstruction algorithm

The image measured by the sensor is unexploitable to the human eye. It is a complex pattern that is the result of the interaction between the scene and the mask. The goal of the reconstruction algorithm is to recover the image of the scene from the sensor measurement. In the general case, this is a complex ill-posed problem whose formulation is given in section 1.3. Approaches to solve this problem are the main focus of this internship.

Reconstruction algorithms need to be able to reconstruct an image close enough to the original in terms of not only mathematical distance (measured by the PSNR or L2 norm for instance) but also perceived similarity and quality (measured by LPIPS [7] or SSIM [8]). Moreover, this must be done in a reasonable amount of time: the reconstruction algorithm must be able to reconstruct an image in a few seconds to be usable in real-time application. At the resolution of current cameras (tens of megapixels), this is a very challenging task. The methods presented here are mainly focused on improving the reconstruction quality with a lesser emphasis on reconstruction speed.

1.3 Reconstruction

For the following, we will name $\mathbf{b}$ the measurements on the sensor and $\mathbf{x}$ the image we want to reconstruct. Under the assumption that the diffuser system is linear and shift-invariant, the effect of the diffusion pattern on the measurements is given by the following equation (ignoring the noise):

$$\mathbf{b} = \mathscr{C}(\mathscr{P} * \mathbf{X}) = CH\mathbf{x}, \tag{1.1}$$

where $\mathscr{P}$ is the point spread function (PSF) of the system and $\mathscr{C}$ is the cropping operator to the sensor size [9]. Examples of PSF of both the DiffuserCam Dataset (see subsection 2.1.1) and a simple piece of tape are shown in figure 1.2 If we consider $\mathbf{x}$ to be the vectorized image, then the convolution and cropping operation can be represented as matrix multiplications by H and C respectively. This model is particularly easy to use as the cropping operation is given by the sensor dimension and the PSF can easily be simulated or measured (as the image of a point source on the sensor). Moreover, it is a good approximation of the real system while being compatible with most optimization algorithms.

Under a Gaussian noise assumption, the maximum likelihood estimator of $\mathbf{x}$ is given by:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}||\mathbf{b} - CH\mathbf{x}||_2^2 \tag{1.2}$$

Due to the multiplexing characteristic of most lensless camera PSFs, this problem is ill-posed and regularization is needed. A traditional approach is to use a non-negativity and a sparsity constraint in the total variation space [9, 10]. This yields the following optimization problem:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}||\mathbf{b} - CH\mathbf{x}||_2^2 + \mathcal{R}(\mathbf{x}) \tag{1.3}$$

which can be solve with traditional optimization algorithm such as ADMM [11], whose pseudocode is shown in algorithm 2.

1.3.1 FISTA

We use a non-negativity constraint for FISTA. The optimization problem is then:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\geq 0} \frac{1}{2}||\mathbf{b} - CH\mathbf{x}||_2^2$$

The FISTA algorithm is an extension of proximal gradient descent that uses a momentum update to accelerate convergence. The algorithm is given in algorithm 1.

Algorithm 1 FISTA

Require: $\mathbf{x}^0 = \mathbf{s}^0$, $\gamma > 0$, and $\tau_1 = 1$
  for $k = 1$ to t **do**
    $\mathbf{z}^k \leftarrow \mathbf{x}^{k-1} - \gamma \Delta g(\mathbf{x}^{k-1})$ ▷ Data fidelity
    $\mathbf{s}^k \leftarrow \text{prox}_{\gamma h}(\mathbf{z}^k)$ ▷ Regularization
    $\tau_{k+1} \leftarrow \frac{1+\sqrt{1+4\tau_k^2}}{2}$
    $\mathbf{x}^k \leftarrow \mathbf{s}_k + \frac{1-\tau_k}{\tau_{k+1}}(\mathbf{s}_k - \mathbf{s}_{k-1})$ ▷ FISTA Update
  end for

1.3.2 ADMM

We use both the non-negativity constraint and total variation as regularization for ADMM. The optimization problem is then:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\geq 0} \frac{1}{2}||\mathbf{b} - CH\mathbf{x}||_2^2 + \tau TV(\mathbf{x})$$

The ADMM algorithm is based on augmented Lagrangian methods, the problem is decomposed into 3 subproblems and their associated dual constraints. The pseudo-code is given in algorithm 2.

1.4 Baseline results

The results from those algorithms are presented in Figure 1.3. In addition to FISTA, results from ADMM with two sets of parameters (default of LenslessPiCam [3] and those used in Monakhova et al. [2]), Gradient Descent (or more precisely proximal gradient descent) and

Algorithm 2 ADMM

Require: $\tau > 0$, $\mu_1 > 0$, $\mu_2 > 0$, $\mu_3 > 0$

for $k = 1$ to t **do**

$u^{k+1} \leftarrow \mathcal{T}_{\tau/\mu_2}\left(\mathbf{\Psi}\mathbf{x}^k + \alpha_2^k/\mu_2\right)$ ▷ Sparsifying soft-threshold

$v^{k+1} \leftarrow \left(\mathbf{C^T C} + \mu_1 I\right)^{-1}\left(\alpha_1^k + \mu_1 \mathbf{H}\mathbf{x}^k + \mathbf{C^T b}\right)$ ▷ least-squares update

$w^{k+1} \leftarrow \max\left(\alpha_3^k/\mu_3 + \mathbf{x}^k, 0\right)$ ▷ enforce non-negativity

$r^k \leftarrow \left(\left(\mu_3 w^{k+1} - \alpha_3^k\right) + \mathbf{\Psi^T}\left(\mu_2 u^{k+1} - \alpha_2^k\right) + \mathbf{H^T}\left(\mu_1 v^{k+1} - \alpha_1^k\right)\right)$

$\mathbf{x}^{k+1} \leftarrow \left(\mu_1 \mathbf{H^T H} + \mu_2 \mathbf{\Psi^T \Psi} + \mu_3 I\right)^{-1} r^k$ ▷ least-squares update

$\alpha_1^{k+1} \leftarrow \alpha_1^k + \mu_1\left(\mathbf{H}\mathbf{x}^{k+1} - v^{k+1}\right)$ ▷ dual for v

$\alpha_2^{k+1} \leftarrow \alpha_2^k + \mu_2\left(\mathbf{\Psi}\mathbf{x}^{k+1} - u^{k+1}\right)$ ▷ dual for u

$\alpha_3^{k+1} \leftarrow \alpha_3^k + \mu_3\left(\mathbf{x}^{k+1} - w^{k+1}\right)$ ▷ dual for w

end for

Algorithm	τ	μ_1	μ_2	μ_3
ADMM	0.0001	10^{-6}	10^{-5}	4×10^{-5}
ADMM_Monakhova	0.002	10^{-4}	10^{-4}	10^{-4}

Table 1.2: Parameters used for each algorithm

Nesterov's accelerated gradient descent are presented. The parameters used for the two versions of ADMM are given in Table 1.2.

Those algorithms are evaluated across a wide range of metrics:

- LPIPS_Alex [7] is a perceptual metric that tries to measure the perceived distance between two images. It exploits the intermediate representation of AlexNet [12] and computes the distance using a multi-layer perceptron on those representations. It is supposed to be more robust to small changes in the image than other perceptual metrics.
- LPIPS_VGG [7] is the same metric as LPIPS_alex but using the intermediate representation of VGG16 [13]. In practice, we found this metric to be more discriminative than LPIPS_alex on very similar images.
- SSIM [8] is a metric that tries to measure the structural similarity between two images. It is based on the luminance, contrast, and structure of the images.
- PSNR is the peak signal-to-noise ratio between the two images.
- MSE is the mean squared error between the reconstructed image and the ground truth.
- ReconstructionError is the mean squared error between the lensless image and the projection to lensless space of the reconstructed image.

As seen in figure 1.3, proximal gradient descent is a lot slower to converge than other metrics, on all metrics except LPIPS it manages to be competitive with other algorithms given enough time. More evolved algorithms with a momentum update (FISTA and Nesterov's gradient descent) are a lot faster to converge and give similar results on most metrics except both LPIPS where they are noticeably better than other methods. Finally, ADMM gives the best results with SSIM, PSNR, MSE and ReconstructionError. However, it is noticeably worse than FISTA and Nesterov gradient descent on both LPIPS metrics. Interestingly, the default parameters of LenslessPiCam give better results on the non-LPIPS metrics than the parameters used in Monakhova et al. [2] while converging faster. They are, however, totally outperformed by the parameters used in Monakhova et al. on both LPIPS metrics.

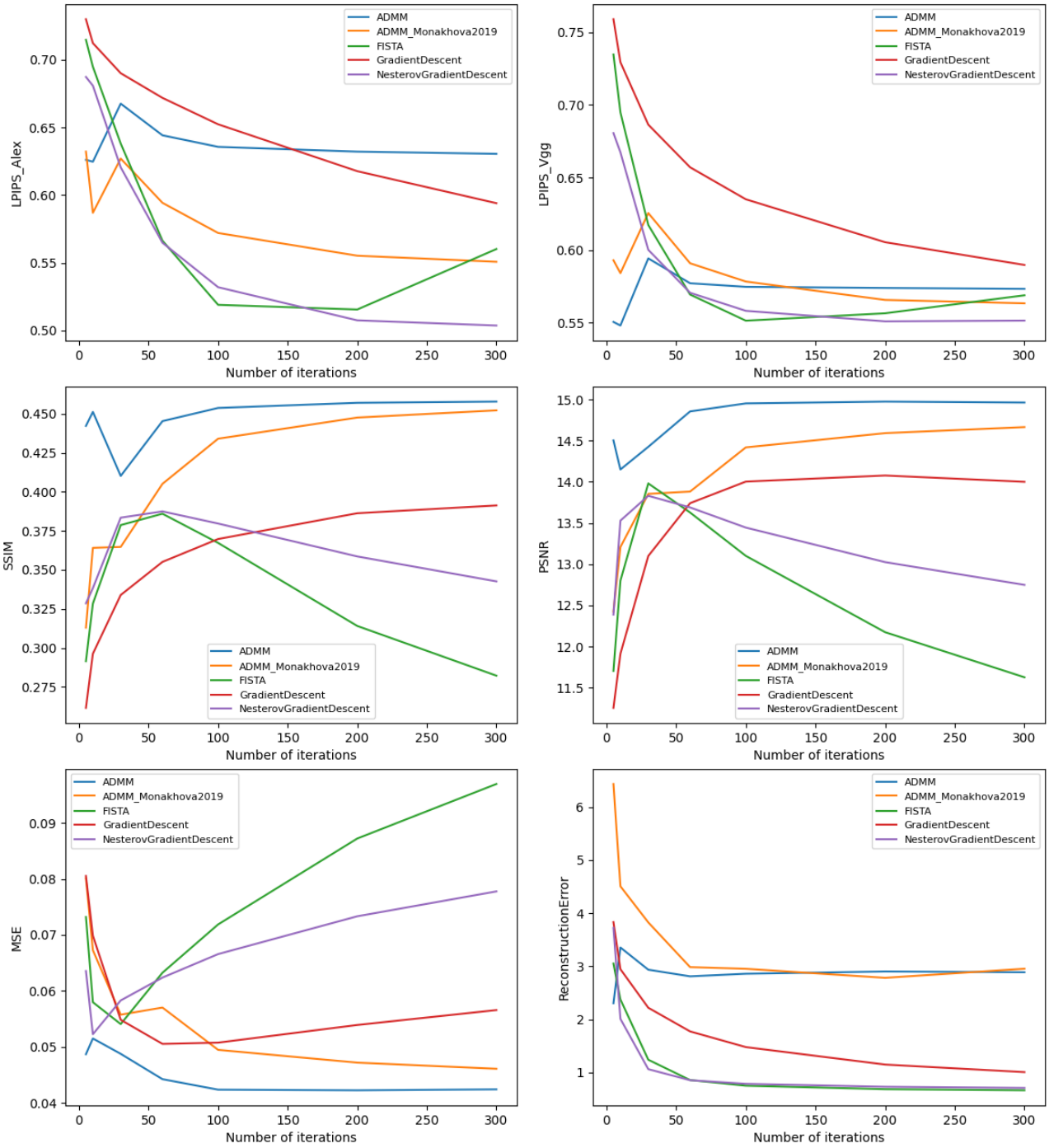

Figure 1.3: Results from classical algorithms on the test set.

1.5 State-of-the-art

Image recovery traditionally relies on iterative convex optimization, aiming to minimize a combined loss function [14, 11]. This loss function comprises two components: a data-fidelity term ensuring alignment of the reconstructed image (modified by a known imaging model) with the actual measurement, and an optional regularization term that incorporates prior knowledge about image attributes to address ill-conditioned challenges. However, these approaches are susceptible to artifacts arising from model disparities, calibration errors, and subjective parameter and regularization choices, collectively diminishing image quality. Moreover, these iterative methods often demand a significant number of iterations, rendering them impractical for real-time imaging scenarios.

Recent trends have seen the emergence of deep learning-based methods for image reconstruction. Utilizing Convolutional Neural Networks (CNNs) [15, 16, 17], these techniques excel in capturing complex scene statistics. However, they lack integration of physical insights into the image formation process, rendering them less interpretable, without guaranteed convergence, and devoid of a systematic approach to incorporate knowledge about the underlying imaging system physics.

A promising middle ground emerges around unrolled optimization. This novel strategy treats a fixed number of iterations from a classic algorithm as a deep network. Each iteration takes on the role of a network layer, permitting the optimization of algorithm parameters that are differentiable with respect to the output, through backpropagation using a specified loss function. This framework also enables learning from training data [18], as demonstrated in diverse applications such as image denoising [19, 20], sparse coding [18]. By synergizing the strengths of deep learning and incorporating physical insights, unrolled optimization enhances both image quality and reconstruction speed, offering a practical solution for everyday imaging challenges, particularly with lensless cameras [2].

1.6 Objectives of the internship

The initial goals of this internship were:

- Replicating and implementing unrolled optimization [2] into LenslessPiCam.
- Investigating how Plug-and-play methods could be apply to lensless imaging.
- Exploring new ideas for lensless reconstruction.
- Investigate how the mask pattern of the lensless camera can be jointly learned with the reconstruction algorithm, in the spirit of deep optics.

Before, the internship the usage of a generator as a prior was discussed. But it was abandoned in favor of exploring new reconstruction algorithms for lensless imaging. Those include Plug-and-play methods which have never been explored in the context of lensless photography before; and a pre-/post- denoising scheme built upon unrolled optimization. The results of those methods are presented in chapter 4 and chapter 5 respectively.

Chapter 2

Experimental setup

2.1 Dataset

2.1.1 Measured Dataset

For all training on real images(all unless specified otherwise), we used the DiffuserCam Lensless Mirflickr Dataset introduced by Monakhova et al. [2]. It consists of 25000 pairs of standard images and their associated lensless image. All images are taken from a screen with a setup involving both a traditional camera and a lensless camera. A beam splitter is used to ensure both cameras capture the exact same image (see figure 2.1). The lensless camera is constructed by placing a piece of glass with variable thickness (phase mask) in front of the sensor, the PSF of the system can be found in figure 1.2.

The dataset contains images of resolution 480 x 270 pixels, already down-sampled from 1920 x 1080 measurements. To reduce the memory requirement for training deep models, we down-sample the images further to a resolution of 240 x 135 pixels. A fortunate side effect is that it makes the moiré from the screen almost invisible.

The first 1000 image pairs from the dataset are taken as a test set, and the 24000 remaining pairs are used to train the model.

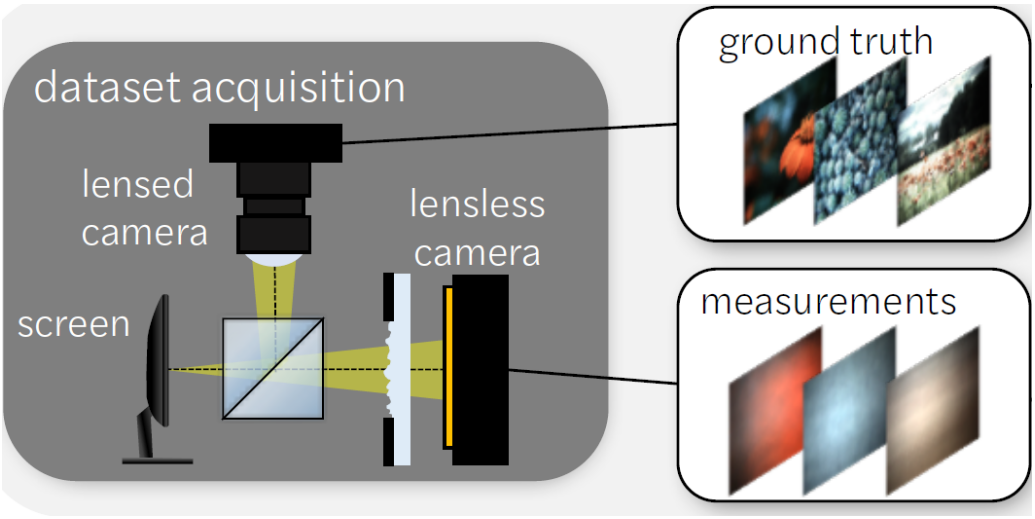

Figure 2.1: Experimental setup for the DiffuserCam Lensless Mirflickr Dataset [2].

SNR(dB)	Loss on simulated	Loss on DC	SSIM on DC	LPIPS (VGG) on DC
00	0.2061	0.3326	0.4489	0.4985
10	0.1344	0.2831	0.6141	0.4215
20	0.0642	0.3251	0.5482	0.4877
40	0.0565	0.3503	0.5241	0.5321

Table 2.1: Unrolled ADMM (20 iter) trained on a simulated dataset and tested on DiffuserCam dataset (DC).

2.1.2 Simulated Dataset

We also use a simulator to create a lensless image corresponding to a given image. The simulator is based on the approximation presented in section 1.3 but it adds Poisson noise and quantifies the result on 8 bits. The simulator is able to convert any traditional single-image dataset to a dataset of image pairs (lensed and lensless). In practice, we mainly use the CelebA dataset [21] for this purpose. The CelebA dataset consists of 202599 images of celebrity faces. 10% of those are used in the test set and the rest in the trainset. The images are downsampled to the PSF size.

To test the effectiveness of this simulated dataset, we train unrolled ADMM (see section 3.2) on the Simulated dataset (using DiffuserCam PSF) and evaluate it on the DiffuserCam dataset. The results with different signal-to-noise ratio (SNR) are presented in Table 2.1. In this experiment an older version of the loss is used where the LPIPS distance is multiplied by a factor of 0.6. This shouldn't affect the results. What we see is that the best result on the test set is obtained for a signal-to-noise ratio of 10. However, even at this value, the loss on the test set is more the double the loss on the trainset. This shows that our simple simulation isn't a very good representation of the real world. What isn't sure is if it is because of our simple noise model or because the hypothesis of section 1.3 doesn't hold in practice.

2.2 Training setup

All experiments are run on a Dell Precision 5820 Tower X-Series (08B1) machine with an Intel i9-10900X CPU and two NVIDIA RTX A5000 GPUs. PyTorch [22] is used for dataset preparation and training. All models are trained for 50 epochs (unless specified otherwise) on the 24000 image pairs with a batch size of 8. The Adam optimizer [23] is used with a learning rate of 10^{-5}. The loss function is the sum of the MSE and the LPIPS [7] distance (using the VGG version) between the output of the model and the ground truth image:

$$\mathscr{L}(y, \hat{y}) = \mathscr{L}_{MSE}(y, \hat{y}) + \mathscr{L}_{LPIPS}(y, \hat{y})$$

The code source including both the models and training code is accessible on GitHub as part of the LenslessPiCam project [3].

Chapter 3

Unrolled optimization

Unrolled optimization is a well-known technique to speed up classical algorithms at the cost of limiting them to a specific domain while also needing a training step to find the parameters of the unrolled algorithm (this requires a dataset containing ground truth). The idea of using unrolled optimization for lensless reconstruction is not new. Monakhova et al. [2] already applied an unrolled version of ADMM with 5 iterations with massive success. They found that the unrolled version of ADMM with 5 iterations gives similar results to classical ADMM with 100 iterations while being 20 times faster. They also found that the unrolled version of ADMM with 5 iterations give better result than classical ADMM with 100 iterations.

As this was done at the beginning of my internship, my goal with unrolled optimization was to reproduce the results obtained by Monakhova et al. in an open-source way (while Monakhova et al. published part of their code, it did not include training). Moreover, in their paper, Monakhova et al. only experimented with ADMM while unrolled could apply to any other algorithm. As a way to check the performance on a more simple algorithm, I also implemented unrolled FISTA.

The main idea behind unrolled algorithms is to realize that each iteration of an algorithm can be seen as a layer of a deep neural like network. The parameters of each layer are the parameters of the algorithm (step size, regularization parameter, etc.) and the input of the layer is the current state of the algorithm after the previous iteration. The output of the layer is the next state of the algorithm. This transforms very little for the algorithm except that parameters vary per iteration. A description of both unrolled FISTA and unrolled ADMM are given in algorithm 3 and 4 respectively.

If the unrolled algorithm is differentiable then it becomes possible to train the algorithm to find the best parameters for each layer. This can be done by backpropagation as usually done in Deep Learning. As long as the algorithm is implemented in a framework supporting autograd (Pytorch in our case), this can be done with very little effort. However, since our data fidelity term includes both Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT), proper support for this algorithm in PyTorch is only possible since version 1.8 released in 2021[1].

The main challenge is then to construct a dataset of lensed images as well as their corresponding lensless measurements. In the case of lensless imaging, Monakhova et al. [2] already captured a dataset (called DiffuserCam) of 25000 images by taking photos of a screen with both a lensed and lensless camera. While the resulting dataset isn't perfect (artifacts from the screen are visible

[1] https://pytorch.org/blog/the-torch.fft-module-accelerated-fast-fourier-transforms-with-autograd-in-pyTorch/

on lensed images and some images are overexposed), they found that their algorithm was still able to generalize to real data. More detail about the dataset is given in subsection 2.1.1.

3.1 Unrolled FISTA

Algorithm 3 Unrolled FISTA

Require: $\mathbf{x}^0 = \mathbf{s}^0$, $\{\gamma_k\}_{k\geq 0}$, and $\{\tau_k\}_{k\geq 0}$

for $k = 1$ to t **do**

$\mathbf{z}^k \leftarrow \mathbf{x}^{k-1} - \gamma_k \Delta g(\mathbf{x}^{k-1})$ ▷ Data fidelity

$\mathbf{s}^k \leftarrow \text{prox}_{\gamma_k h}(\mathbf{z}^k)$ ▷ Regularization

$\mathbf{x}^k \leftarrow \mathbf{s}_k + \frac{1-\tau_k}{\tau_{k+1}}(\mathbf{s}_k - \mathbf{s}_{k-1})$ ▷ FISTA Update

end for

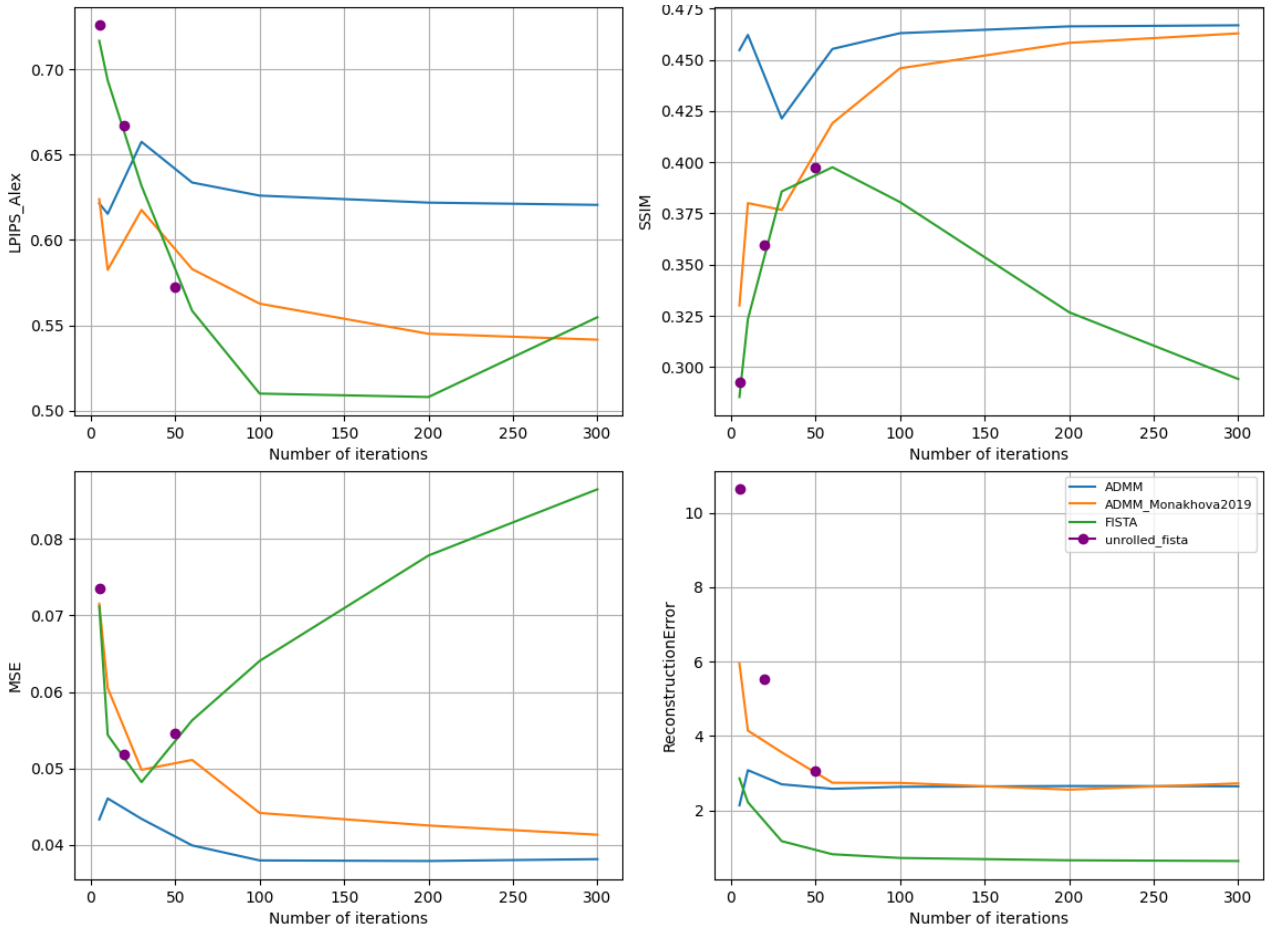

Figure 3.1: Results from unrolled FISTA trained on DiffuserCam.

Implementation-wise, the PyTorch library allows for automatized differentiation of all the operations used in the algorithm. The only difficulty in the implementation is to ensure proper support for operation on a batch of images to speed up training (by 4 times compared to a naive implementation with a batch of a single image).

Moreover, because of the computational cost of the FFT and IFFT used to convolve with the PSF, each layer of the algorithm is a lot more costly than a simple convolution layer with a small kernel(in both time but also memory). This limit severely, the resolution of the image that can be used during training. Working directly on HD images is out of the question. Even the low

default resolution of the DiffuserCam dataset (see subsection 2.1.1) can be challenging to fit in VRAM (GPU memory) for the deepest version of the unrolled algorithm. Therefore, all the results presented here are measured on a down-sampled version of the image (around 100×200). This difficulty is present for the training of all algorithms based on unrolled (chapter 3 and also chapter 5).

The results of unrolled FISTA are presented in Figure 3.1. Unrolled FISTA performs only marginally better if not worse than traditional FISTA on most metrics. This result shows how well-optimized FISTA is and how hard it is to improve directly on it. However, compared to the other method explored during this internship, unrolled FISTA is limited by its very simple non-negativity constraint and lack of a better regularization.

3.2 Unrolled ADMM

Algorithm 4 Unrolled ADMM

Require: $\{\tau_k\}_{k\geq 0}$, $\{\mu_1^k\}_{k\geq 0}$, $\{\mu_2^k\}_{k\geq 0}$, $\{\mu_3^k\}_{k\geq 0}$

for $k = 1$ to t **do**

$u^{k+1} \leftarrow \mathcal{T}_{\tau_k/\mu_2^k}\left(\mathbf{\Psi}\mathbf{x}^k + \alpha_2^k/\mu_2^k\right)$ ▷ Sparsifying soft-threshold

$v^{k+1} \leftarrow \left(\mathbf{C^T C} + \mu_1^k I\right)^{-1}\left(\alpha_1^k + \mu_1^k \mathbf{H}\mathbf{x}^k + \mathbf{C^T b}\right)$ ▷ least-squares update

$w^{k+1} \leftarrow \max\left(\alpha_3^k/\mu_3^k + \mathbf{x}^k, 0\right)$ ▷ enforce non-negativity

$r^k \leftarrow \left(\left(\mu_3^k w^{k+1} - \alpha_3^k\right) + \mathbf{\Psi^T}\left(\mu_2^k u^{k+1} - \alpha_2^k\right) + \mathbf{H^T}\left(\mu_1^k v^{k+1} - \alpha_1^k\right)\right)$

$\mathbf{x}^{k+1} \leftarrow \left(\mu_1^k \mathbf{H^T H} + \mu_2^k \mathbf{\Psi^T \Psi} + \mu_3^k I\right)^{-1} r^k$ ▷ least-squares update

$\alpha_1^{k+1} \leftarrow \alpha_1^k + \mu_1^k\left(\mathbf{H}\mathbf{x}^{k+1} - v^{k+1}\right)$ ▷ dual for v

$\alpha_2^{k+1} \leftarrow \alpha_2^k + \mu_2^k\left(\mathbf{\Psi}\mathbf{x}^{k+1} - u^{k+1}\right)$ ▷ dual for u

$\alpha_3^{k+1} \leftarrow \alpha_3^k + \mu_3^k\left(\mathbf{x}^{k+1} - w^{k+1}\right)$ ▷ dual for w

end for

Once again, the implementation is very easy from a theoretical point of view. The only difference with classical ADMM is that μ_1, μ_2, μ_3, and τ are now learnable parameters with different values for each iteration of the algorithm. The only real difficulty is to ensure proper support for batchs of images. This is even more true here as the algorithm is a lot more complex than unrolled FISTA, particularly for the total variation and its adjoint.

However, the results are a lot more interesting compared to unrolled FISTA (see Figure 3.2). Unrolled ADMM is able to outperform classical ADMM on the similarity metrics (LPIPS and SSIM) by an impressive margin at the cost of a small increase in MSE and an important increase in reconstruction error. This is expected considering that the training process doesn't take into account the reconstruction error at all. Figure 3.3 show some examples of the result of unrolled ADMM compared to classical ADMM with 20 iterations. The images obtained with Unrolled ADMM are a lot sharper and with more details compared to ADMM. This is particularly clear on top of the cake where the floral motifs are visible only in the unrolled version. However, unrolled ADMM also introduces some artifacts in the image in the form of horizontal or vertical lines.

When compared to the result from Monakhova et al., my implementation of unrolled ADMM is better on all metrics except LPIPS with 5 iterations (same number of iterations as they use). However, we see massive gains when increasing the number of iterations to 20 and 50. This allows us to outperform their result on all metrics. Since Monakhova et al. didn't clearly publish the loss use during training, we can expect most of the difference to be the result of a bigger emphasis on the LPIPS distance in the loss.

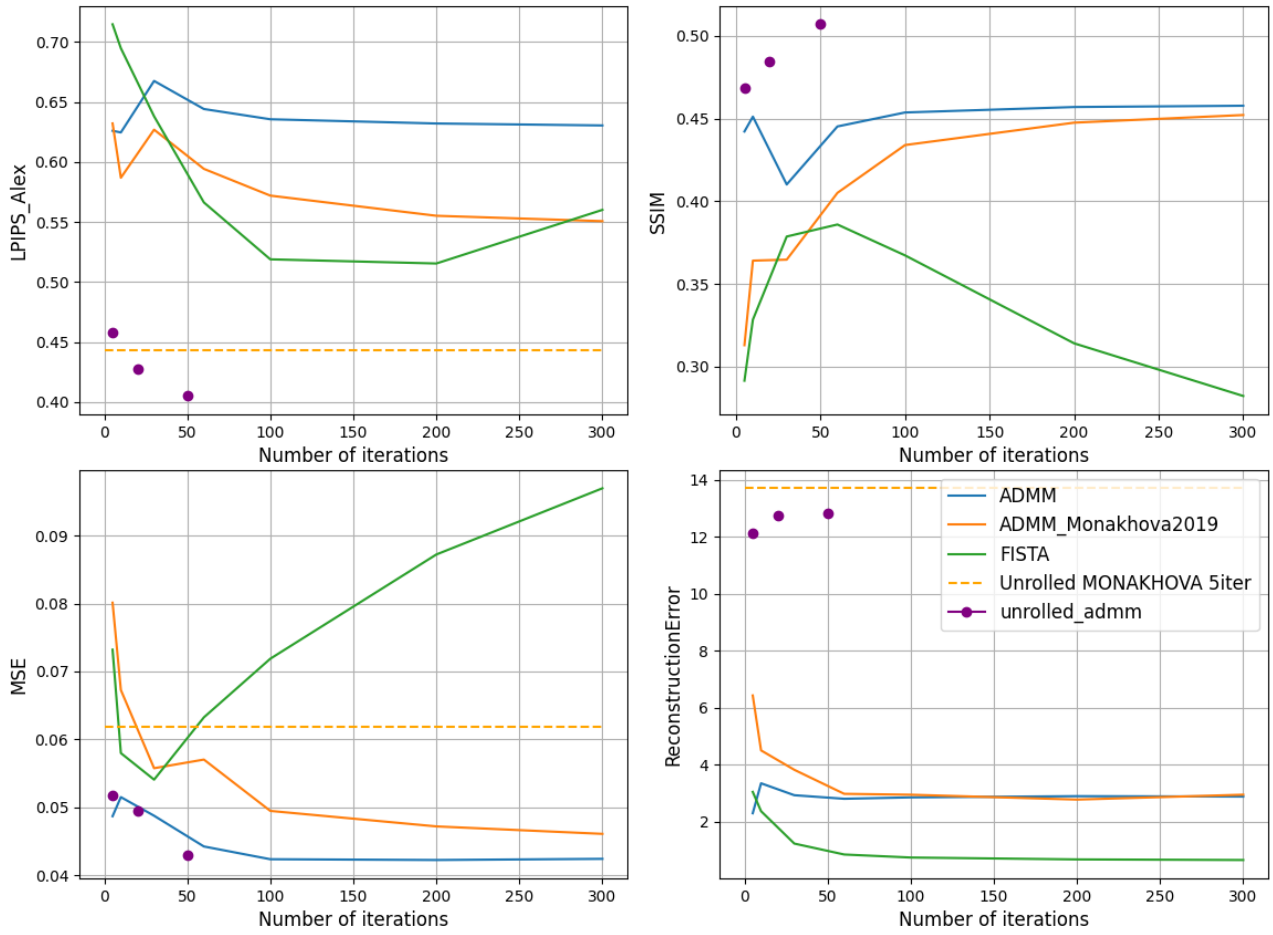

Figure 3.2: Results from unrolled ADMM trained on DiffuserCam.

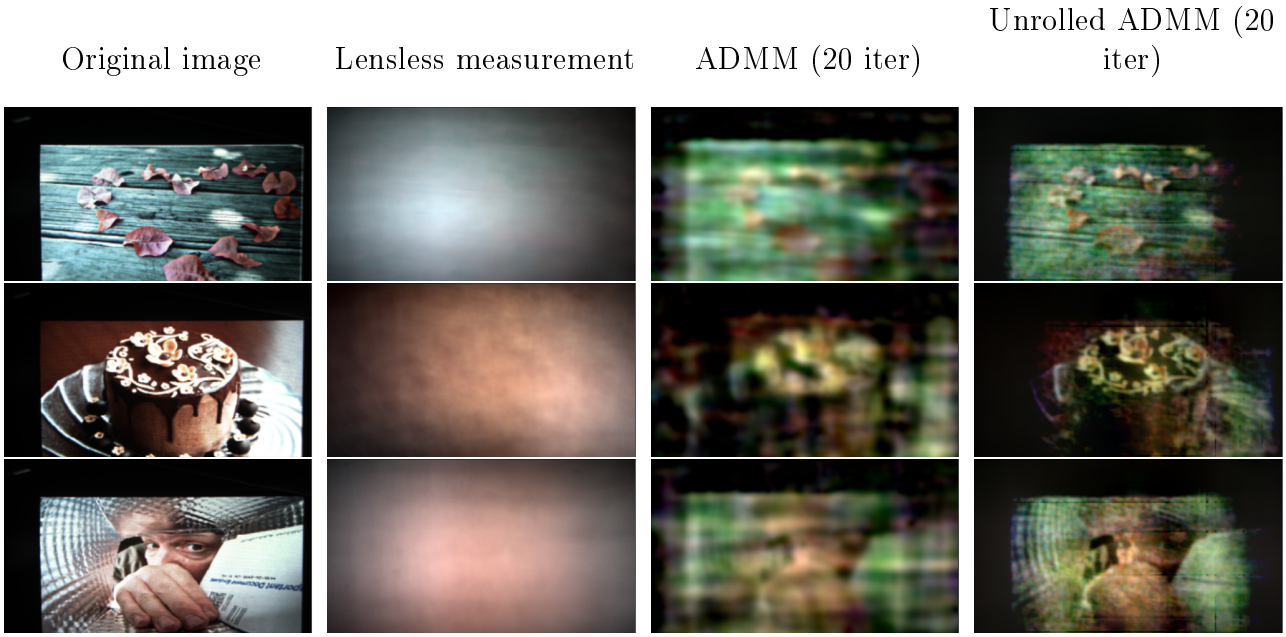

Figure 3.3: Result of unrolled ADMM compared to ADMM

Chapter 4

Plug-and-play method

A recent class of algorithm that has shown great success in inverse problems is the Plug-and-play method [24]. The main idea is to replace the proximal operator of the regularization ($\mathcal{T}_{\tau/\mu_2}$ in algorithm 2) by a denoiser. The resulting algorithm doesn't necessarily solve an optimization problem anymore, however, this approach has been shown to give good results in practice. Usually, neural denoisers that are able to constrain the solution to a space much closer to the natural image space, are used.

While Plug-and-play approach has been used in the context of coherent diffractive imaging for microscopie [25], we are not aware of any work that uses this approach in the context of natural light lensless photography.

4.1 FISTA Plug-and-play

The implementation of FISTA Plug-and-play is relatively simple. The proximal operator of the regularization is replaced by a DRUNet denoiser [26] (An alternative with BM3D was also explored but abandoned because of the long computation time). There is no need to make sure the algorithm is differentiable since no training is required.

The only difficulty is to estimate the best value of the noise to be used. We tried both a constant noise value for all iterations as well as an exponentially decreasing noise value. After searching for optimal start and stop values (only for the exponentially decreasing variant), we found that the best results were obtained with a constant noise value of 0.01. The results are presented in Figure 4.1.

Algorithm 5 FISTAPnP

Require: $\mathbf{x}^0 = \mathbf{s}^0$, $\gamma > 0$, and $\tau_1 = 1$
  for $k = 1$ to t **do**
    $\mathbf{z}^k \leftarrow \mathbf{x}^{k-1} - \gamma \Delta g(\mathbf{x}^{k-1})$ ▷ Data fidelity
    $\mathbf{s}^k \leftarrow \mathcal{D}_\sigma(\mathbf{z}^k)$ ▷ Regularization
    $\tau_{k+1} \leftarrow \frac{1+\sqrt{1+4\tau_k^2}}{2}$
    $\mathbf{x}^k \leftarrow \mathbf{s}_k + \frac{1-\tau_k}{\tau_{k+1}}(\mathbf{s}_k - \mathbf{s}_{k-1})$ ▷ FISTA Update
  end for

The results are really underwhelming, The Plug-and-play approach is only slightly better than FISTA for less than 40 iterations before completely falling off. Moreover, this small gain in performance comes at the cost of a massive increase in computation time, from 48ms per image

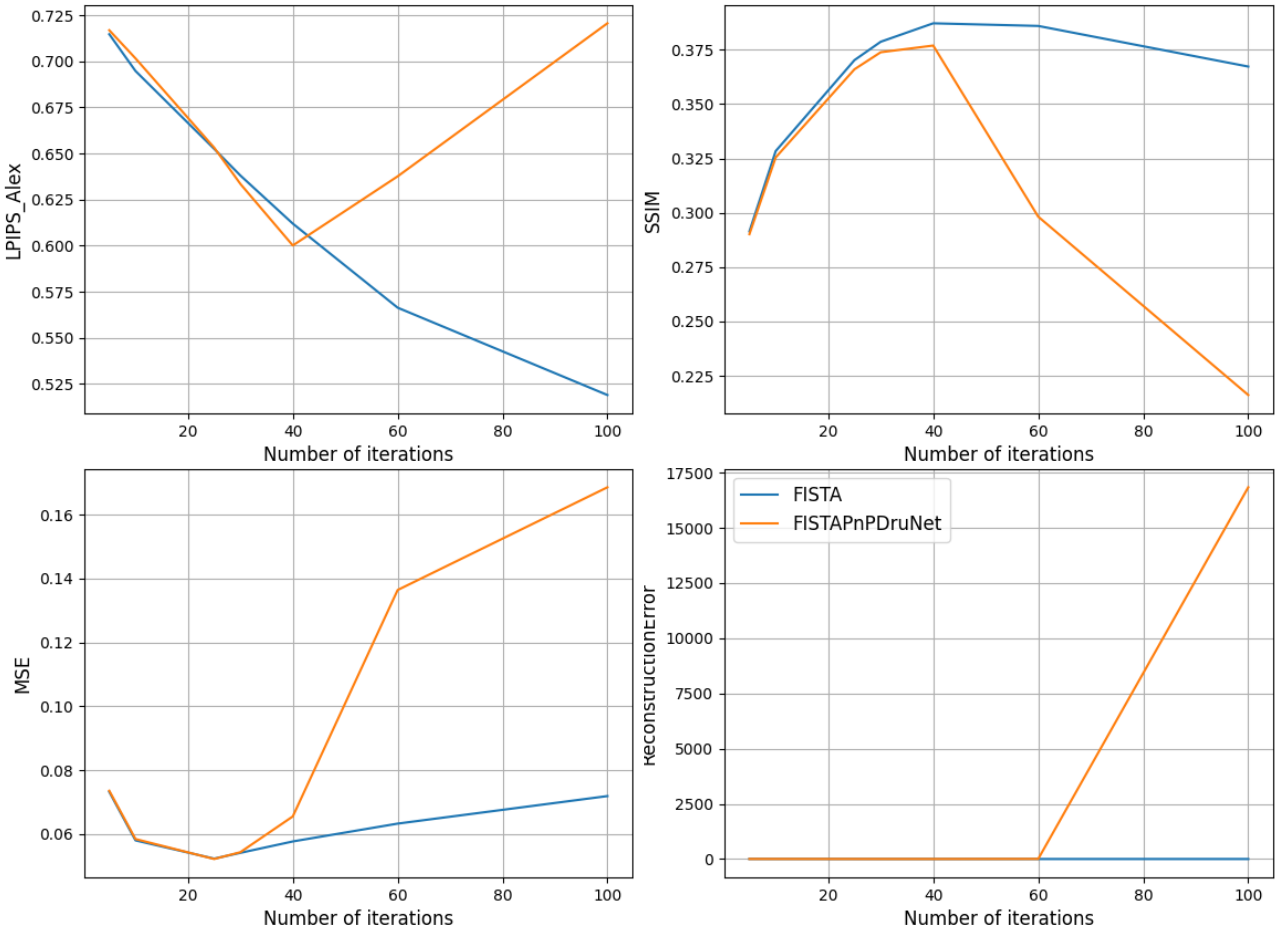

Figure 4.1: Results of Plug-and-play FISTA evaluated on DiffuserCam.

to 1s per image at 100 iterations. This is due to the fact that the denoiser is a lot more costly than the proximal operator of the regularization.

4.2 ADMM Plug-and-play

The implementation of ADMM Plug-and-play is a bit more complicated than FISTA. The proximal operator of the regularization is replaced by a DRUNet denoiser [26], however, this raises the question of how it should be incorporated into the dual update system. Our implementation removes the associated dual variable as described in algorithm 6. This is, however, quite a liberal way of inserting the denoiser. A more conservative way would be to replace the proximal operator of the regularization with the denoiser and keep the dual variable, but we found this approach to be less performant.

We used a similar approach to estimate the best value of the noise to be used. We tried both a constant noise value for all iterations as well as an exponentially decreasing noise value. We obtained the best result with a constant noise value of 10. The results are presented in Figure 4.2.

The results here are a lot more interesting. For less than 40 iterations, ADMM Plug-and-play is able to outperform traditional ADMM on LPIPS. However, this once again comes at the cost of longer inference time (from 50ms to 6s for 100 iterations).

Figure 4.3 illustrates the effect of Plug-and-play on ADMM. For 20 iterations the resulting image is a lot sharper than the one from traditional ADMM. However, the denoiser struggles

Algorithm 6 ADMMPnP

Require: $\tau > 0$, $\mu_1 > 0$, $\mu_2 > 0$, $\mu_3 > 0$

for $k = 1$ to t **do**

$u^{k+1} \leftarrow \mathcal{D}_\sigma\left(\mathbf{x}^k\right)$ ▷ Sparsifying soft-threshold

$v^{k+1} \leftarrow \left(\mathbf{C^T C} + \mu_1 I\right)^{-1}\left(\alpha_1^k + \mu_1 \mathbf{H}\mathbf{x}^k + \mathbf{C^T b}\right)$ ▷ least-squares update

$w^{k+1} \leftarrow \max\left(\alpha_3^k/\mu_3 + \mathbf{x}^k, 0\right)$ ▷ enforce non-negativity

$r^k \leftarrow \left(\left(\mu_3 w^{k+1} - \alpha_3^k\right) + \mu_2 u^{k+1} + \mathbf{H^T}\left(\mu_1 v^{k+1} - \alpha_1^k\right)\right)$

$\mathbf{x}^{k+1} \leftarrow \left(\mu_1 \mathbf{H^T H} + \mu_2 I + \mu_3 I\right)^{-1} r^k$ ▷ least-squares update

$\alpha_1^{k+1} \leftarrow \alpha_1^k + \mu_1\left(\mathbf{H}\mathbf{x}^{k+1} - v^{k+1}\right)$ ▷ dual for v

$\alpha_3^{k+1} \leftarrow \alpha_3^k + \mu_3\left(\mathbf{x}^{k+1} - w^{k+1}\right)$ ▷ dual for w

end for

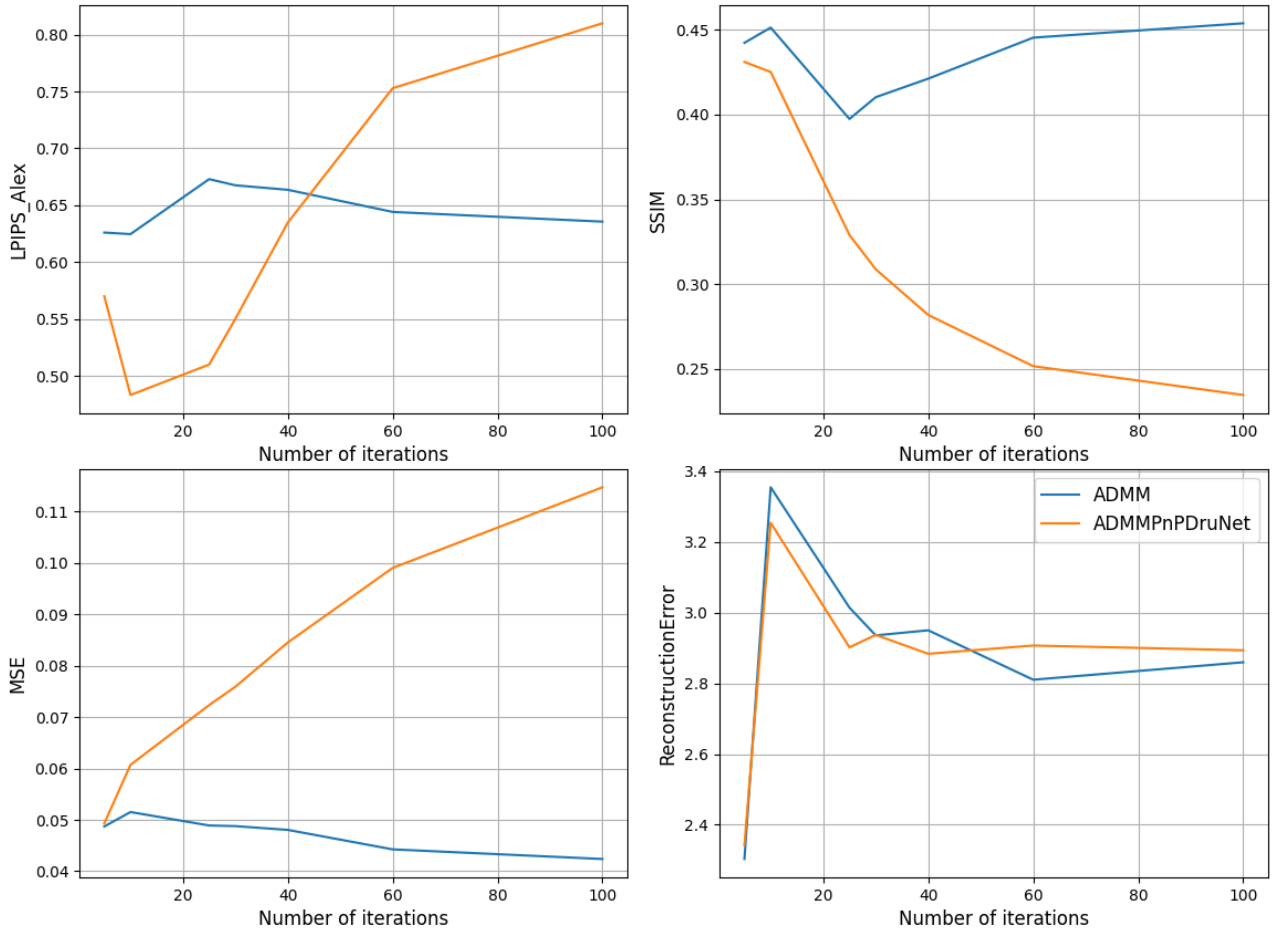

Figure 4.2: Results of Plug-and-play ADMM evaluated on DiffuserCam.

to remove the noise associated with the reconstruction algorithm, and the resulting image still has a considerable amount of artifacts from the reconstruction. For 100 iterations, the resulting image is almost pointillist. The image seems to have been oversharpen by the denoiser and the resulting image is a lot worse than the one from traditional ADMM. We try to reduce this effect by using a partial update scheme or lower noise level as input to the denoiser without much success. More advanced regularization techniques such as Regularization by Denoising [27] could help solve this problem but were not tested.

While PnP is on all metrics worse than unrolledADMM (see 3.2), it still brings a massive perceptual improvement compared to ADMM alone which seems hard to measure even with similarity metrics like LPIPS. This naturally raises the question of whether or not this approach

could be compatible with unrolled to bring even bigger improvement. However, this combined approach would need the DRUNet model to be loaded in memory once for each iteration of the algorithm, making it prohibitively expensive. As an alternative, we propose a pre- and post-denoising scheme that only use two DRUNet-like model for the reconstruction.

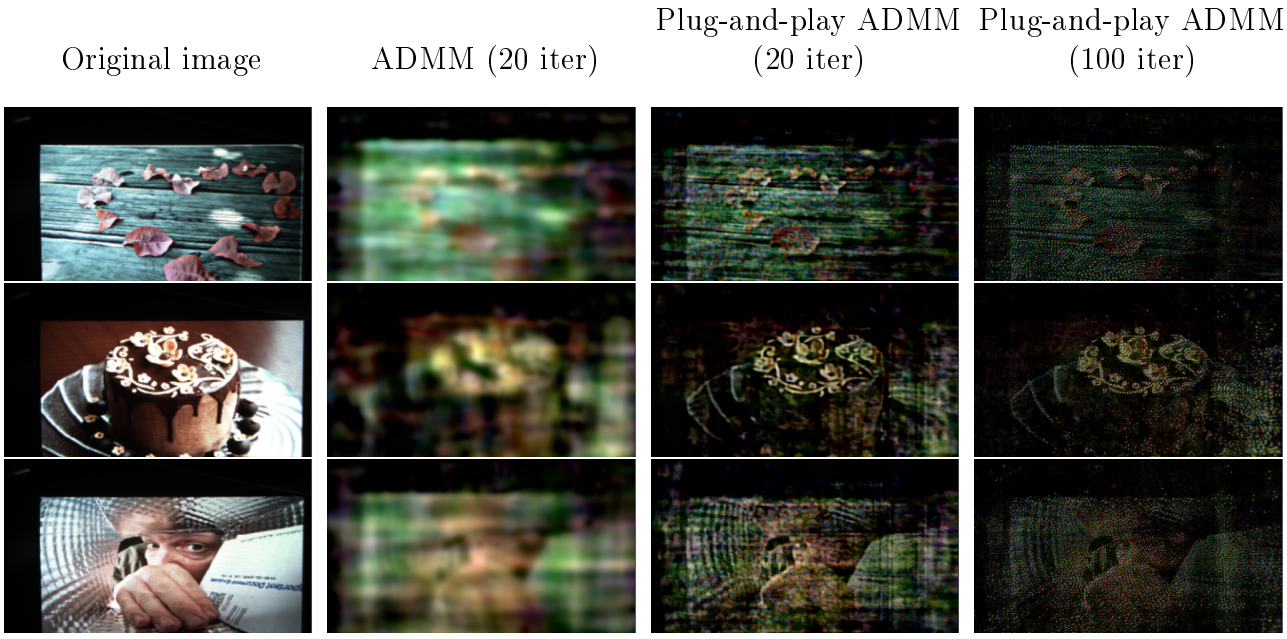

Figure 4.3: Result of Plug-and-play ADMM compare to traditional ADMM

Chapter 5

Pre/post-denoising

Previous works [2, 10] show that the combination of an inversion "layer" based on traditional algorithms before a deep convolutional network is beneficial. We go one step further and propose a dual denoising approach(Figure 5.1) with a pre-denoiser to reduce the measurement noise and handle the unique artifact of the lensless system and, a post-denoiser to reduce the artifact from the reconstruction algorithm. To achieve optimal results, the post-denoiser should be able to correct the color balance, sharpen the image, and more.

Both denoisers have a very challenging task. The pre-denoiser works with a well-known noise (sensor noise), but measurements lack any of the structure usually used by most denoisers. On the contrary, the post-denoiser works in the well-known space of natural images, but the artifacts from the reconstruction algorithm are complex with very high spatial coherency. Our solution to this problem is to train both denoisers and the reconstruction algorithm end-to-end as a single entity. The training process follows the parameter described in chapter 2.

Figure 5.1: Proposed denoising pipeline

5.1 Models

All models use unrolled ADMM as the central algorithm with 5 to 20 iterations(see section 3.2). The main tradeoff here is of quality to execution time. While our results show that the quality of the reconstructed images increases with the number of iterations, the execution time of the reconstruction algorithm is the main bottleneck of the whole pipeline. This is due to the high cost of the convolution in the forward operator.

The denoiser networks are U-Net with d residual blocks between each down- and up-sampling layer. There are three down-sampling layers using stride convolution with a stride of two and three up-sampling layers using transposed convolution. The number of channels is 64 for the first layer and is doubled at each down-sampling layer. For $d = 4$, the resulting architecture is identical to the one of DRUNet [26](see figure 5.2). The number of parameters is around $d \times 10^7$ for each denoiser network. We found that while increasing the size of the denoiser networks improves massively the quality of the reconstruction at the cost of a small increase in

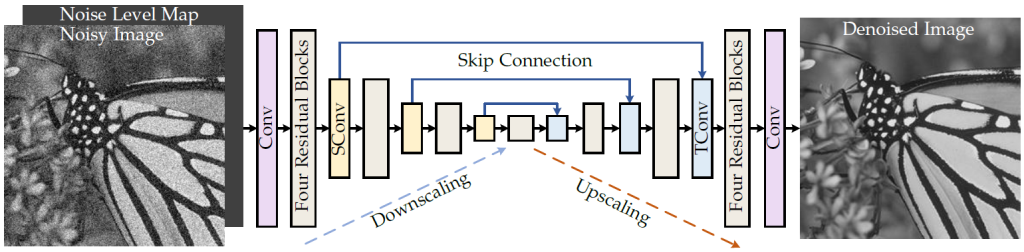

Figure 5.2: DRUNet architecture [26]

Reconstruction algorithm	Pre-denoiser	Post-denoiser	PSNR	LPIPS(Vgg)	M
ADMM (100 iter)	None	None	14.95	0.4618	0.
Unrolled ADMM (5 iter)	None	None	12.93	0.4833	0.
Unrolled ADMM (20 iter)	None	None	13.11	0.4541	0.
PnP ADMM (20 iter)	None	None	12.8	0.486	0.
Unrolled ADMM (20 iter)	None	Fine-tuned DRUNet ($d = 4$)	22.80	0.1979	0.
Unrolled ADMM (20 iter)	UNetRes ($d = 2$)	None	17.42	0.2887	0.
Unrolled ADMM (20 iter)	UNetRes ($d = 2$)	UNetRes ($d = 2$)	24.50	0.1675	0.
Unrolled ADMM (5 iter)	UNetRes ($d = 2$)	Fine-tuned DRUNet ($d = 4$)	24.77	0.1231	0.
Unrolled ADMM (20 iter)	UNetRes ($d = 2$)	Fine-tuned DRUNet ($d = 4$)	**25.64**	**0.1106**	**0.**

Table 5.1: Result of different model on the test set.

execution time, this doesn't work past a certain point (maybe due to our limited dataset). We were not able to train a denoiser network with more than 2 residual blocks (except while using finetuning).

5.2 Results

Table 5.1 summarized the results obtained with different models, and Figure 5.4 compares reconstructions of a few of the approaches on images from the test set. The best result is obtained with a UNetRes with $d = 2$ as a pre-denoiser (20M parameters) and a full DRUNet fine-tuned as post denoiser (40M parameters). This approach brings an improvement of 12dB of PSNR and 4x of LPIPS compared to a simple unrolled ADMM with no pre- or post-denoiser (third row in Table 5.1 and second row in Figure 5.4). Even compared to just unrolled ADMM with a post-denoiser (architecture similar to Le-ADMM-U proposed in [2]) the improvement are massive with better colors and more detail (see Figure 5.4). The result of the best model on a few examples from the test set are shown in Figure 5.3 with the intermediary output of the different stage.

Our experiment shows that for a similar number of parameters, using a mixed approach with both a pre-denoiser and a post-denoiser performed better than using all those parameters as a post-denoiser as proposed in [2]. This is very apparent in Figure 5.4. The model using two UNetRes (d=2) is not only able to reconstruct the colors more accurately but also able to reconstruct more detail (particularly visible on the hand). Figure 5.3 shows that the pre-denoiser is not just denoising the image, even adding in new artifacts in the result of unrolled ADMM. However, those are a lot more structured and easier to remove for the post-denoiser. Moreover, we can see that there is a massive color shift in the images after the reconstruction algorithm. This is surprising as we have a different PSF for each color channel, however as shown in figure 1.2, the PSF for each color channel are very similar. It seems that in practice, with this PSF, there is little distinction made between the different color channels.

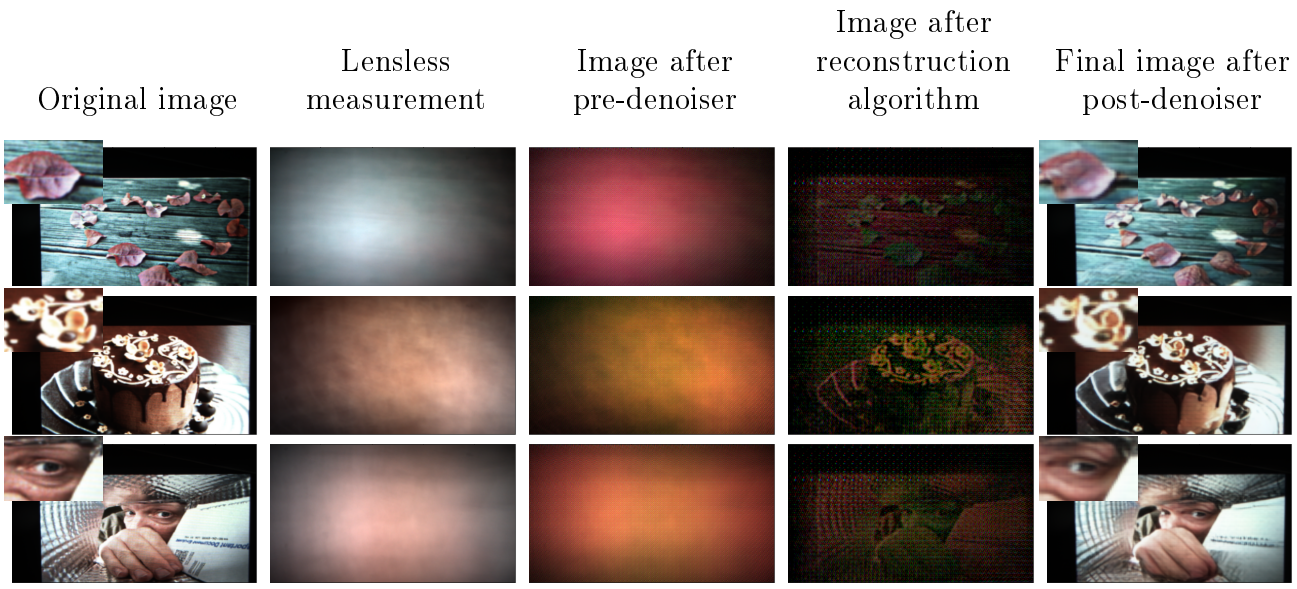

Figure 5.3: Result of unrolled ADMM with a UNetRes with 20M parameters as pre-denoiser and a full DRUNet fine-tunned as post-denoiser.

As expected, an increase in the size of the three parameters (number of iterations, number of parameters in the pre-processor, and number of parameters in the post-processor) results in a direct improvement of the image quality metrics. Despite our limited training set size, fine-tuning a DRUNet denoiser [26] (already trained on natural images) as the post-processor allows us to use a bigger network and hence obtain a noticeable improvement in performance. An alternate approach to improving the reconstruction without requiring significantly more data is to increase the number of unrolled iterations of ADMM.

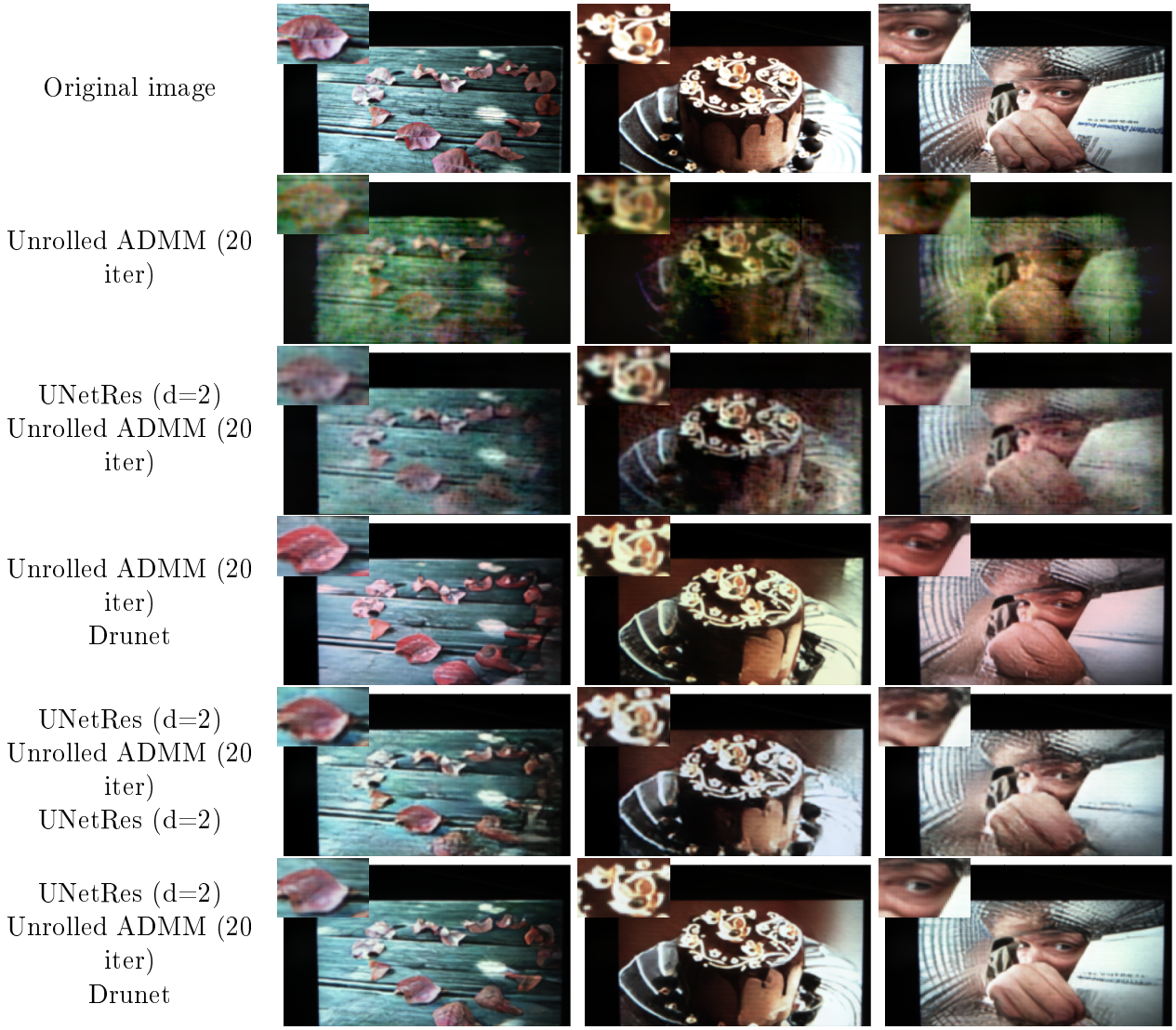

Figure 5.4: Comparative result of different combinations of pre-/post-denoiser.

Chapter 6

Learning the PSF

During the last few weeks of my internship, I developed a framework allowing to optimize the PSF. However, the space of acceptable PSF is limited by the type of mask and the manufacturability of those masks. The main difficulty is to limit the search space in a way that allows easy optimization through gradient descent.

The idea is to parametrize the mask with parameters θ_{mask} such that the PSF can be computed through a differentiable function f:

$$\mathscr{P} = f(\theta_{mask}) \tag{6.1}$$

f is supposed to compute the propagation of the light from a single point source through the mask. θ_{mask} is then optimized using gradient descent using the loss function described in 2.2. To favor sparsity, L1 regularization can be used on θ_{mask}. Conditions can easily be imposed on θ_{mask} using a projected gradient schema. Figure 6.1 show how this learnable mask can be integrated with the different reconstruction algorithm. Using a measure dataset such as the DiffuserCam Lensless Mirflickr Dataset, it is possible to find the optimal mask for the reconstruction algorithm, this is explored in section 6.1. However, it is possible to go even further using a simulated dataset and optimize the PSF from both capture and reconstruction. This is explored in section 6.2. For both cases, f is the identity function, meaning that we are directly learning the PSF with for sole constraint that $\mathscr{P} \in [0,1]^{n,m,3}$.

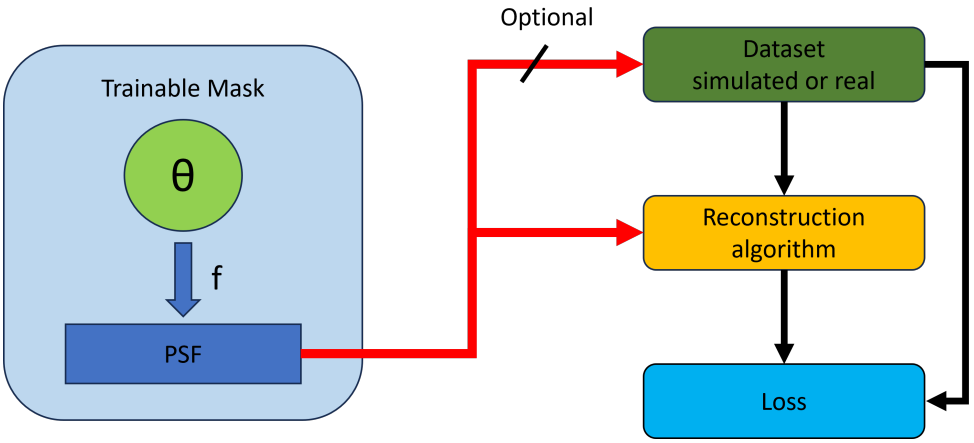

Figure 6.1: Trainable mask

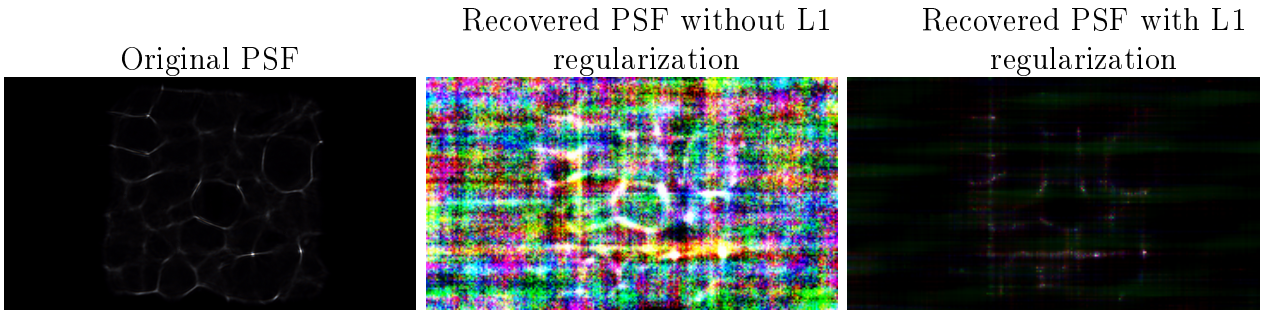

Figure 6.2: Result of PSF recovery from DiffuserCam.

PSF	PSNR	LPIPS	MSE
Original PSF	13.11	**0.4541**	0.04955
Recovered PSF without L1 regularization	12.61	0.5893	0.05570
Recovered PSF with L1 regularization	**14.98**	0.5947	**0.03252**

Table 6.1: Comparison of different PSF recovery from DiffuserCam.

6.1 PSF from measurement

In the simplest form, it is possible to train a mask (or PSF in this case) from an existing dataset. This means, only the gradient from the reconstruction algorithm is used to optimize the PSF. Ideally, the result should be the original PSF of the dataset.

To test this algorithm, we use the DiffuserCam dataset and unrolled ADMM with 20 iterations. Training is done for 50 epochs with ADAM optimizer [23] and a learning rate of 0.001 for the mask. The PSF is randomly initialized with a uniform value in $[0, 1]^{n,m,3}$. We do not use LPIPS in the loss as we struggle to obtain any result when using it. The results are shown in figure 6.2 with and without L1 regularization for the PSF. Even after 50 epochs, the recovered PSF without regularization is still very noisy (although the original PSF started to emerge). The regularized PSF is a lot closer to the original. Interestingly, it also gives a better reconstruction result (see table 6.1) according to PSNR and MSE. However, the LPIPS (which we didn't optimize for) is a lot worse, and, in practice, this results in a more blurry and noisy reconstructed image.

Another option is to start from the correct PSF and to fine-tune it to improve the reconstruction. We use the same reconstruction algorithm as before but with LPIPS in the loss (as it doesn't cause any problem for fine-tuning) and only fine-tuned for 10 epochs. The resulting PSF is shown in figure 6.3. While the PSF fine-tuned without L1 regularization has accumulated a lot of noise, the one with L1 regularization has stayed close to the original PSF. The results in Table 6.2 show that both fine-tuned PSFs give a better PSNR and MSE than the original PSF. However, only the PSF fine-tuned with a regularizer improves the LPIPS. The improvement is massive with 2dB of PSNR and 0.08 of LPIPS. Considering the low number of parameters (32 400), this is an impressive result.

PSF	PSNR	LPIPS	MSE
Original PSF	13.11	0.4541	0.04955
Fine-tuned PSF without L1 regularization	15.31	0.5556	**0.03017**
Fine-tuned PSF with L1 regularization	**15.31**	**0.3759**	0.03060

Table 6.2: Comparison of different PSF fine-tuned from DiffuserCam.

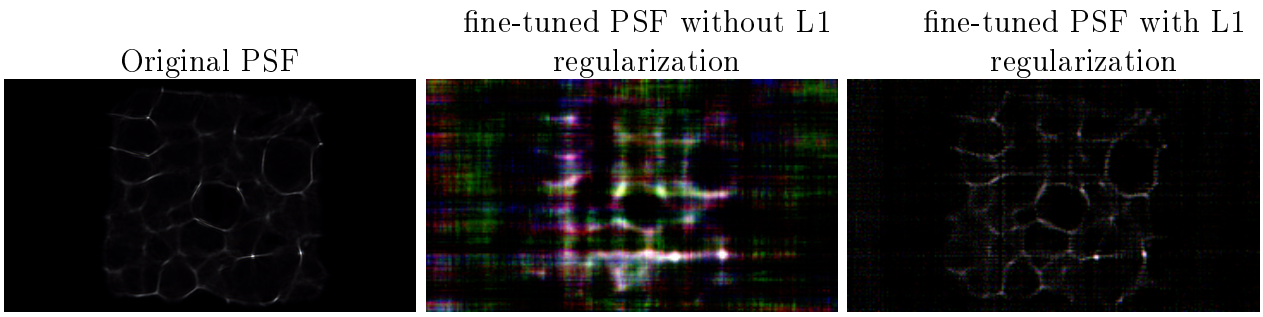

Figure 6.3: Result of finetuning a PSF for DiffuserCam.

6.2 Learning an optimal PSF

Another possibility offered by this trainable mask framework is to optimize the mask for both the measurement and the reconstruction. This work is still in a very early stage. We directly learn the PSF without any of the constraints from chapter 6 using unrolled ADDM with 20 iterations as the reconstruction algorithm using a differentiable simulated dataset obtained from the CelebA dataset. Considering the result obtained in section 6.1, we use L1 regularization for all the experiments. We found that using a different PSF for each color channel always results in a single color dominating the output. Therefore, we use a single-channel PSF which is applied similarly to all color channels.

The results are shown in table 6.3. While the idea is interesting, this very simple implementation clearly doesn't work. The optimized PSF is worse than the DiffuserCam PSF on all metrics. The optimized PSF for a noise level of 10dB is shown in Figure 6.4. Just from the regular structure, one would expect this PSF to give subpar results. Clearly, more work needs to be done to constrain the mask and better guide the optimization towards better results.

PSF	Simulation SNR	PSNR	LPIPS(VGG)	MSE
DiffuserCam PSF	10dB	**13.11**	**0.4541**	**0.04284**
Optimized PSF	10dB	10.27	0.6220	0.09685
DiffuserCam PSF	40dB	**21.84**	**0.1283**	**0.006606**
Optimized PSF	40dB	21.20	0.1561	0.007693

Table 6.3: Performance of the optimized PSF for different noise levels.

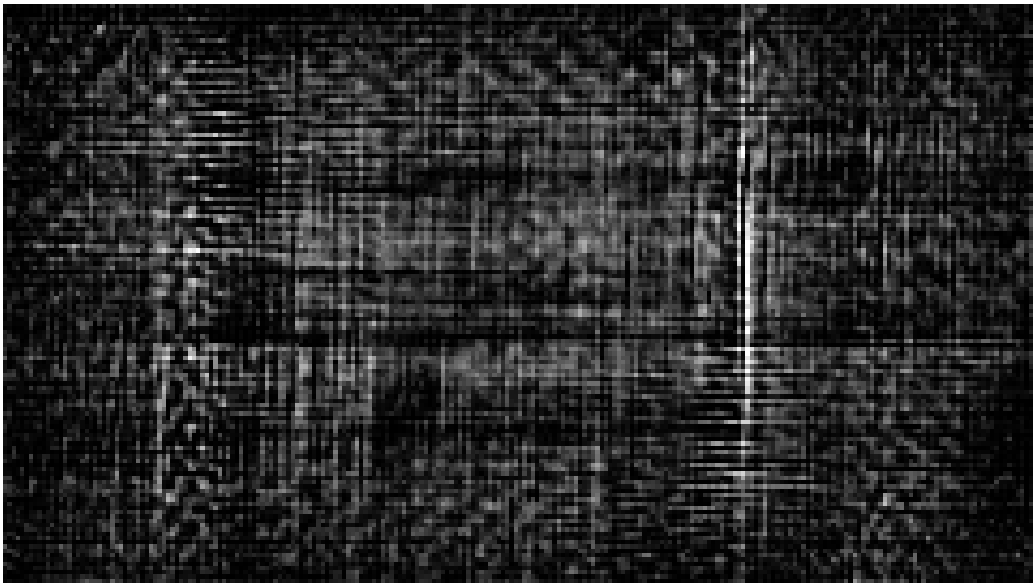

Figure 6.4: Optimized PSF for a noise level of 10dB.

Chapter 7

Conclusion

During my internship, I worked on multiple reconstruction algorithms for lensless imaging. Those include the unrolled versions of both ADMM and FISTA that are able to generalize to arbitrary numbers of iterations (With a good scaling of the quality), a Plug-and-play version of ADMM with visible improvement of the perceived quality at the cost of a much longer algorithm, and a pre-/post-denoiser architecture which achieve state-of-the-art result on all metrics, often giving result indistinguishable from the ground truth. All those algorithms have been made open source as part of the LenslessPiCam project [1] [3].

I also worked on a framework allowing to optimize the mask or the PSF of the system. This framework is able to optimize the PSF for either or both the measurement and the reconstruction. This has shown promising results while fine-tuning the PSF to increase reconstruction quality. However, the optimization of the PSF from scratch is still in a very early stage and more work on the constraints and optimization are needed to obtain good results. Moreover, the simulation framework currently used for optimizing with respect to the measurement has been shown to generalize poorly to real measurement. This puts into question the real-world performance of any optimization with respect to the measurement.

The main challenge that should be addressed for lensless imaging in the future is scaling to higher resolutions. Even on a high-end GPU, the unrolled algorithm (with or without pre- and post-denoiser) takes around 50ms on a 240x135 image. This is already too slow for real-time applications and is expected to grow linearly with the number of pixels. This renders HD reconstruction on the camera itself unpractical even for modern high-end smartphones. Moreover, from my limited experiment, I found that the reconstruction quality tends to decrease as the resolution of the reconstructed image increases. This may be due to the artifacts resulting from the DiffuserCam dataset capture setup being more visible at higher resolution, or it might arise from a potential inherent inability of lensless cameras to capture information with high frequencies (even with a high-resolution measurement). If it is the second, one could expect the PSF to be the reason, and more work on the mask itself would be needed to improve the reconstruction quality at high resolution.

[1] https://github.com/LCAV/LenslessPiCam

Bibliography

[1] Haoran You, Cheng Wan, Yang Zhao, Zhongzhi Yu, Yonggan Fu, Jiayi Yuan, Shang Wu, Shunyao Zhang, Yongan Zhang, Chaojian Li, Vivek Boominathan, Ashok Veeraraghavan, Ziyun Li, and Yingyan Lin. EyeCoD. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*. ACM, jun 2022.

[2] Kristina Monakhova, Joshua Yurtsever, Grace Kuo, Nick Antipa, Kyrollos Yanny, and Laura Waller. Learned reconstructions for practical mask-based lensless imaging. *Opt. Express*, 27(20):28075–28090, Sep 2019.

[3] Eric Bezzam, Sepand Kashani, Martin Vetterli, and Matthieu Simeoni. Lenslesspicam: A hardware and software platform for lensless computational imaging with a raspberry pi. *Journal of Open Source Software*, 8(86):4747, 2023.

[4] M. Salman Asif, Ali Ayremlou, Ashok Veeraraghavan, Richard Baraniuk, and Aswin Sankaranarayanan. Flatcam: Replacing lenses with masks and computation. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 663–666, 2015.

[5] Kazuyuki Tajima, Takeshi Shimano, Yusuke Nakamura, Mayu Sao, and Taku Hoshizawa. Lensless light-field imaging with multi-phased fresnel zone aperture. In *2017 IEEE International Conference on Computational Photography (ICCP)*, pages 1–7, 2017.

[6] Patrick R. Gill, James Tringali, Alex Schneider, Salman Kabir, David G. Stork, Evan Erickson, and Mark Kellam. Thermal escher sensors: Pixel-efficient lensless imagers based on tiled optics. In *Imaging and Applied Optics 2017 (3D, AIO, COSI, IS, MATH, pcAOP)*, page CTu3B.3. Optica Publishing Group, 2017.

[7] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CoRR*, abs/1801.03924, 2018.

[8] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[9] Nick Antipa, Grace Kuo, Reinhard Heckel, Ben Mildenhall, Emrah Bostan, Ren Ng, and Laura Waller. Diffusercam: Lensless single-exposure 3d imaging. *CoRR*, abs/1710.02134, 2017.

[10] Vivek Boominathan, Jesse Adams, Jacob Robinson, and Ashok Veeraraghavan. Phlatcam: Designed phase-mask based thin lensless camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 04 2020.

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.* 2011.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[15] Shuai Li, Mo Deng, Justin Lee, Ayan Sinha, and George Barbastathis. Imaging through glass diffusers using densely connected convolutional networks. *Optica*, 5(7):803–813, Jul 2018.

[16] Shuai Li, Mo Deng, Justin Lee, Ayan Sinha, and George Barbastathis. Imaging through glass diffusers using densely connected convolutional networks. *Optica*, 5(7):803–813, Jul 2018.

[17] Thanh Nguyen, Yujia Xue, Yunzhe Li, Lei Tian, and George Nehmetallah. Deep learning approach for fourier ptychography microscopy. *Opt. Express*, 26(20):26470–26484, Oct 2018.

[18] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, 2010.

[19] Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. Unrolled optimization with deep priors, 2018.

[20] Steven Diamond, Vincent Sitzmann, Frank Julca-Aguilar, Stephen Boyd, Gordon Wetzstein, and Felix Heide. Dirty pixels: Towards end-to-end image processing and perception, 2021.

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[24] Ulugbek S. Kamilov, Charles A. Bouman, Gregery T. Buzzard, and Brendt Wohlberg. Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications. *IEEE Signal Processing Magazine*, 40(1):85–97, jan 2023.

[25] Li Song and Edmund Y. Lam. Fast and robust phase retrieval for masked coherent diffractive imaging. *Photon. Res.*, 10(3):758–768, Mar 2022.

[26] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 06 2021.

[27] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red), 2017.