# Physics-Informed Machine Learning for Computational Imaging

*Kristina Monakhova*

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-177
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-177.html

July 29, 2022

# Physics-Informed Machine Learning for Computational Imaging

by

Kristina Monakhova

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Laura Waller, Chair
Associate Professor Ren Ng
Professor Joshua Bloom

Summer 2022

Physics-Informed Machine Learning for Computational Imaging

Abstract

Physics-Informed Machine Learning for Computational Imaging

by

Kristina Monakhova

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Laura Waller, Chair

A key aspect of many computational imaging systems, from compressive cameras to low light photography, are the algorithms used to uncover the signal from encoded or noisy measurements. Some computational cameras encode higher-dimensional information (e.g. different wavelengths of light, 3D, time) onto a 2-dimensional sensor, then use algorithms to decode and recover this high-dimensional information. Others capture measurements that are extremely noisy, or degraded, and require algorithms to extract the signal and make the images usable by people, or by higher-level downstream algorithms. In each case, the algorithms used to decode and extract information from raw measurements are critical and necessary to make computational cameras function. Over the years the predominant methods, *classic methods*, to recover information from computational cameras have been based on minimizing an optimization problem consisting of a data term and hand-picked prior term. More recently, deep learning has been applied to these problems, but often has no way to incorporate known optical characteristics, requires large training datasets, and results in black-box models that cannot easily be interpreted. In this dissertation, we present *physics-informed machine learning* for computational imaging, which is a middle ground approach that combines elements of classic methods with deep learning. We show how to incorporate knowledge of the imaging system physics into neural networks to improve image quality and performance beyond what is feasible with either classic or deep methods for several computational cameras. We show several different ways to incorporate imaging physics into neural networks, including algorithm unrolling, differentiable optical models, unsupervised methods, and through generative adversarial networks. For each of these methods, we focus on a different computational camera with unique challenges and modeling considerations. First, we introduce an unrolled, physics-informed network that improves the quality and reconstruction time of lensless cameras, improving these cameras and showing photorealistic image quality on a variety of scenes. Building up on this, we demonstrate a new reconstruction network that can improve the reconstruction time for compressive, single-shot 3D microscopy with spatially-varying blur by 1,600×, enabling interactive previewing of the scene. In cases where training data is hard to acquire, we show that an

untrained physics-informed network can improve image quality for compressive single-shot video and hyperspectral imaging without the need for training data. Finally, we design a physics-informed noise generator that can realistically synthesize noise at extremely high-gain, low-light settings. Using this learned noise model, we show how we can push a camera past its typical limit and take photorealistic videos at starlight levels of illumination for the first time. Each case highlights how using physics-informed machine learning can improve computational cameras and push them to their limits.

To Viktor and Lubov in the heavens

# Contents

# List of Figures

# List of Tables

# Acknowledgments

I came to grad school thinking I would study robotics. But soon after I realized that I was much more interested in cameras, sensors, and how we perceive the world. First and foremost, I want to thank my advisor, Laura Waller, for taking a chance on me as PhD student when I came to her during my first year, knowing nothing about optics, cameras, or optimization. I really appreciate the time and space Laura gave me to learn all this new material and find my own direction within her lab.

I also want to thank the many professors who I've interacted with through classes, seminars, and meetings throughout the years. I'd like to especially thank Kannan Ramchandran and Reinhard Heckel who first taught me about compressive sensing, Martin Wainwright who taught me optimization, and of course Laura for teaching me optics. Thank you to my quals committee member, Chunlei Liu, and my dissertation committee members Ren Ng and Joshua Bloom for their discussions and useful feedback.

I want to thank the Waller Lab members for the many fun discussions, coffee and boba breaks, and office gossip throughout the years. I was fortunate to be a part of a large cohort of new Waller Lab students who made the first few years of classes, prelims, and adjusting to grad school more enjoyable - so thank you to Kyrollos, Linda, Gautam, David, and Stuart. A special thank you to Grace for all the informal mentorship on both research and life throughout the years. And to Nick, for teaching me what a lens is during my first year of grad school. Thank you to Emma for your mentorship, fun spider stories, and opening up doors for me. I want to especially thank Grace, Regina, Emma, Linda, and Mindy for some wonderful lunches pre-pandemic. And to Guanghan, Henry, Kyrollos, Eric, Ruiming, Neerja, Amit, Charles, Loza, Tiffany, and Leyla who were some of the first and only people I saw in person after the pandemic during some wonderful outdoor group lunches. Thanks to the rest of the Waller Lab members, Yi, Shwetadwip, Emrah, Michael, Li-Hao, Zack, and Herbert for great discussions over the years. I want to especially thank Kyrollos for being the best co-author and grad school friend anyone could ask for - I could not have done it without our frequent conversations, gossip, and pair programming sessions. We said that between the two of us, we could probably get one PhD, but in the end we both made it and got two.

I want to also thank my many wonderful EECS colleagues who have made grad school fun over the years. Thank you to Sarah for being an amazing housemate, friend, and the best wedding officiant. Thank you to Regina, Emma, Mindy, Sarah, Justin, Stan, Michael, and Emily for some fun hiking and backpacking trips. And thank you to Jaimie, Alan, Laura, and Sylvia for great conversations. I want to also thank the incredible Argentine Tango community in Berkeley for giving me a mental break from grad school and providing a source of joy, fun, and friendship throughout the years. Thank you to my tango instructors Homer and Cristina, and to my tango friends, especially Chi, Suzanne, Stephanie, Geoff, Zeke, Jonathan, Justin, Teresa, and Kyra.

Of course, a special thank you to my mother, who was a whole coast away but made time for frequent FaceTime calls. My mother has always been a source of support and strength for me, showing me that anything is possible if you work hard. And to my little brother who

I first met before I moved to Berkeley and who somehow grew up into a 1st grader by the time I finished my PhD.

Finally, thank you to my best friend and life partner Kevin for filling my life with so much joy and happiness - enough happiness to make it though a long, isolating PhD in the middle of a global pandemic. I could not have done it without you.

# Chapter 1

# Introduction

Imaging systems allow us to measure and understand the world. They are critical enablers of science, allowing us to see and study anything from the smallest sub-cellular structures (microscopes) to distant galaxies (telescopes). We use imaging systems to document our daily lives (cameras) and get medical diagnostics (MRI, X-Ray, CT). We send imaging systems up into space to monitor crop health and deforestation (hyperspectral cameras) and to peer into the furthest depths of time (James Webb Space Telescope). If we can make imaging systems that can see smaller, further, in more dimensions, that are faster, cheaper, and smarter, there are boundless possibilities in what we can discover and do.

To improve imaging systems and push them past their native limits, we can use computational imaging. Computational imaging is the co-design of algorithms and camera hardware (e.g. optics, sensors) to make more capable imagers. Traditional cameras are direct imagers, in which the image that is captured is very close to the desired image—meaning that little to no post-processing is required after the image is captured, Fig. 1.1(a). Alternatively, a computational camera uses its hardware to 'encode' information about the scene into the measurement. The raw measurement may look very little like the scene, but all the information is there and algorithms can subsequently be used to extract, and 'decode' the object of interest, Fig. 1.1(b).

It has been shown that we can encode and extract additional information into computational cameras, such as light-fields [162, 125], 3D [123, 8], hyperspectral [72, 130, 209], polarization [203], and temporal information [90, 137]. In addition, computational cameras can be designed to be small, compact, and cheap, such as by removing the bulky lens component and replacing it with a low-cost diffuser [8, 114, 131]. By designing the algorithms together with the optics, we can encode the most useful information for a specific task, then leverage priors and clever algorithms to extract the most information from our measurements.

A key part of computational cameras' pipelines are *inverse problems*, which is how we can reconstruct unknown signals and objects from our encoded measurements. Often times, the operations applied by computational cameras are non-invertible. Recovering the encoded information may be ill-posed, or underdetermined, making it challenging to recover a unique solution [83, 35]. Throughout the years, a number of *classic methods* have been developed

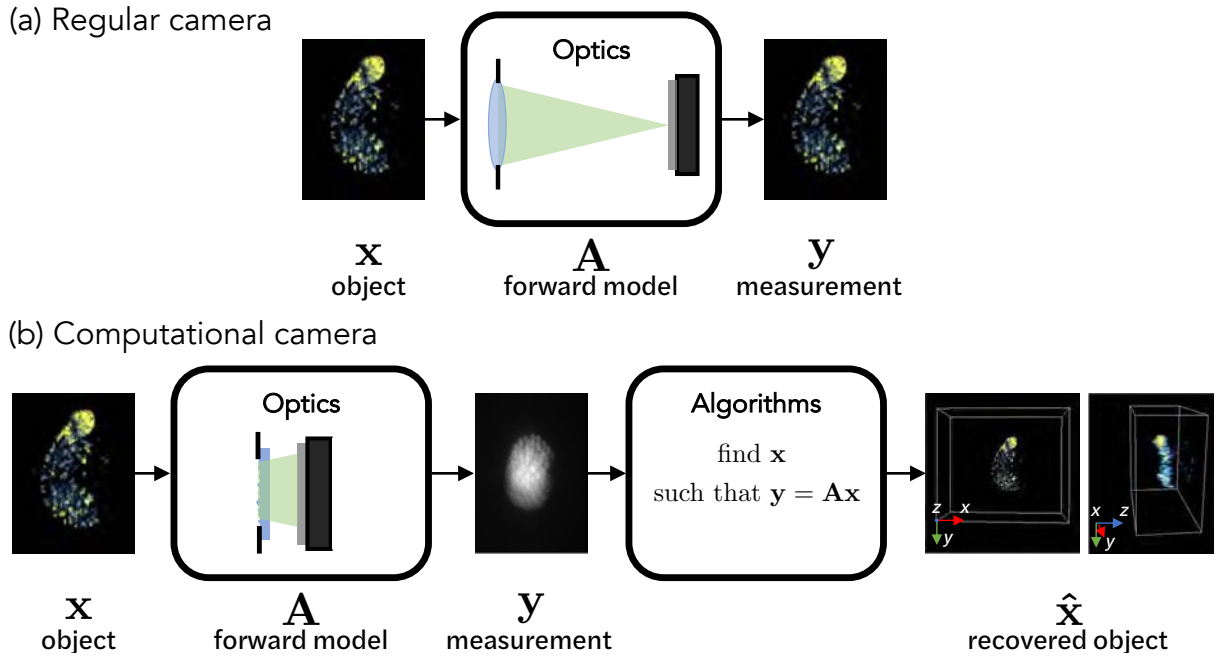


Figure 1.1: **Traditional vs. computational cameras**. (a) A traditional camera directly encodes information, requiring little to no post-processing. (b) A computational camera encodes information from the scene, then uses algorithms to decode and extract information. The process of going from a measurement to a recovered object is called solving an inverse problem.

to solve these imaging inverse problems by using optimization techniques which minimize a data-fidelity term and a hand-designed prior based on the image statistics [52, 20, 28]. More recently, deep learning-based approaches have shown significant improvements and surpassed classic methods at many imaging tasks [127, 128, 163]. Neural networks have hundreds of thousands of parameters which are updated using large datasets of image pairs [119]. These networks are able to learn complex scene statistics, but do not incorporate any prior knowledge about the image formation process. Compared to classic iterative methods, deep learning-based methods are hard to interpret, do not have convergence guarantees, and have no structured way to incorporate knowledge of the imaging system physics.

In this dissertation, we introduce and analyze *physics-informed machine learning* in which we leverage both our knowledge of the imaging system physics together with deep learning to create better algorithms for computational cameras. Using differentiable optical models, we incorporate knowledge of the image formation process into the neural networks, forming physics-informed layers. We show significant improvement over classic and deep methods for a variety of challenging imaging tasks, from compressive single-shot 3D and hyperspectral imaging, to low-light video denoising. We showcase how blending the physics of the imaging system with neural networks can help our algorithms converge faster, improve quality, and

provide more interpretability.

## 1.1 Dissertation Outline

In this dissertation, we outline several different ways in which we can incorporate our knowledge of imaging system physics into neural networks to better solve imaging inverse problems. We will look at both supervised and unsupervised learning methods, as well as methods that require real experimental datasets vs. methods that can generalize to real systems based on simulated data. For each computational camera, there are unique requirements that make certain physics-informed machine learning techniques more applicable than others. Throughout the dissertation, we build up the foundation of physics-informed machine learning, showing how we can push computational cameras to their limits.

**Chapter 2:** This chapter provides relevant background necessary to understand the contributions of this dissertation. We introduce imaging forward models and inverse problems. Next, we discuss the different ways we can solve imaging inverse problems, including classic methods, deep methods, as well as an intro to physics-informed machine learning methods.

**Chapter 3:** In this chapter, we dive into the modeling for compressive lensless cameras. We first build intuition for how we can encode additional information (hyperspectral, time, 3D) into a computational camera and how we can use compressive sensing to recover this information. Next, we go into detail for how to build one of these compressive cameras and all of the design considerations that go into this, focusing on the hyperspectral imager Spectral DiffuserCam. We exclusively look at classic methods in this chapter, giving a sense of how well computational cameras can perform without physics-informed machine learning models.

**Chapter 4:** In this chapter, we introduce a supervised method of physics-informed machine learning for 2D lensless imaging. Our approach is based on algorithm unrolling and uses a real dataset of experimentally captured lensed and lensless image pairs. We show how combining physics-based modeling with data-driven reconstructions can improve the image quality and reconstruction time for lensless cameras.

**Chapter 5:** We extend our notion of physics-informed machine learning to compressive single-shot 3D microscopy. Microscopes often have spatially-varying aberrations, requiring computationally expensive reconstruction algorithms to undo this field-varying blur. We introduce a physics-informed reconstruction network that uses multiple differentiable Wiener filters paired with a neural network refinement step to perform underdetermined spatially-varying deconvolution. This method is a supervised learning technique, but uses entirely simulated data and generalizes to real experimental data. With our network, we achieve a 1,600× speedup in reconstruction times compared to classic methods, while maintaining good resolution and image quality for single-shot 3D microscopy.

**Chapter 6:** In this chapter, we focus on a physics-based unsupervised learning technique. For many computational cameras, it is difficult to obtain a large dataset with ground truth data. We demonstrate an unsupervised reconstruction approach for single-shot video and hyperspectral imaging that does not require training data. We use an untrained network

as an imaging prior while solving an underdetermined inverse problem. We show that our untrained network serves as a better prior than several classic priors, resulting in better image quality.

**Chapter 7:** We introduce a learned, physics-informed noise model for low light photography. In high gain, low light settings, noise is often non-Gaussian and hard to characterize. We show how to use a small dataset of clean/noisy image pairs to train a generative adversarial network to synthesize realistic noise in this regime. Our noise generator consists of a few physics-informed parameters which are tuned from the data, essentially learning a noise forward model from data. This noise model can subsequently be used to generate realistic training data for low-light cameras. We use this noise model to push a camera designed for low light to its limits, demonstrating photorealistic videography in submillilux levels of illumination (starlight).

**Chapter 8:** Finally, we reflect on the themes in this dissertation and discuss a vision for the future of computational cameras.

# Chapter 2

# Background

This chapter provides background on modeling computational cameras and solving inverse problems. In this chapter, we build the foundations for understanding how to mathematically model linear imaging systems and how to solve imaging inverse problems. We introduce three types of methods for solving inverse problems: classic, deep, and physics-informed machine learning, with the latter being the focus of this dissertation.

## 2.1 Forward Model: Modeling Cameras as Linear Systems

First, we need to understand how light from the world goes through our imager and gets recorded by the sensor. This abstraction is called the forward model of the imaging system and models the behavior of the optics, filters, sensor (e.g. quantization), or any other components within the system. We assume that we can describe this behavior as a linear system of equations, in which our measurement, $\mathbf{y}$ is a matrix-vector multiplication with our system matrix, $\mathbf{A}$, and our image intensity, $\mathbf{x}$:

$$\underbrace{\mathbf{y}}_{\text{measurement}} = \mathbf{A} \underbrace{\mathbf{x}}_{\text{object}}. \tag{2.1}$$

In the presence of noise, $\mathbf{n}$, this model becomes:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n} \tag{2.2}$$

A traditional camera consists of a sensor and a lens. The sensor records light hitting it. Without a lens, light from the world hits the sensor from all different objects and directions, resulting in an averaged measurement of the light intensity, Fig. 2.1(a). A lens focuses light from the world, forming an image of the world on the sensor, Fig. 2.1(b). Given a perfect lens, the system matrix is the identity operator, $\mathbf{A} = \mathbf{I}$. For this idealized camera, $\mathbf{y} = \mathbf{x}$, and no further computation is needed.

For an idealized camera in the presence of noise, we have the following forward model: $\mathbf{y} = \mathbf{x} + \mathbf{n}$, Fig. 2.1(c). For this sort of camera, the image is corrupted by noise. This is a

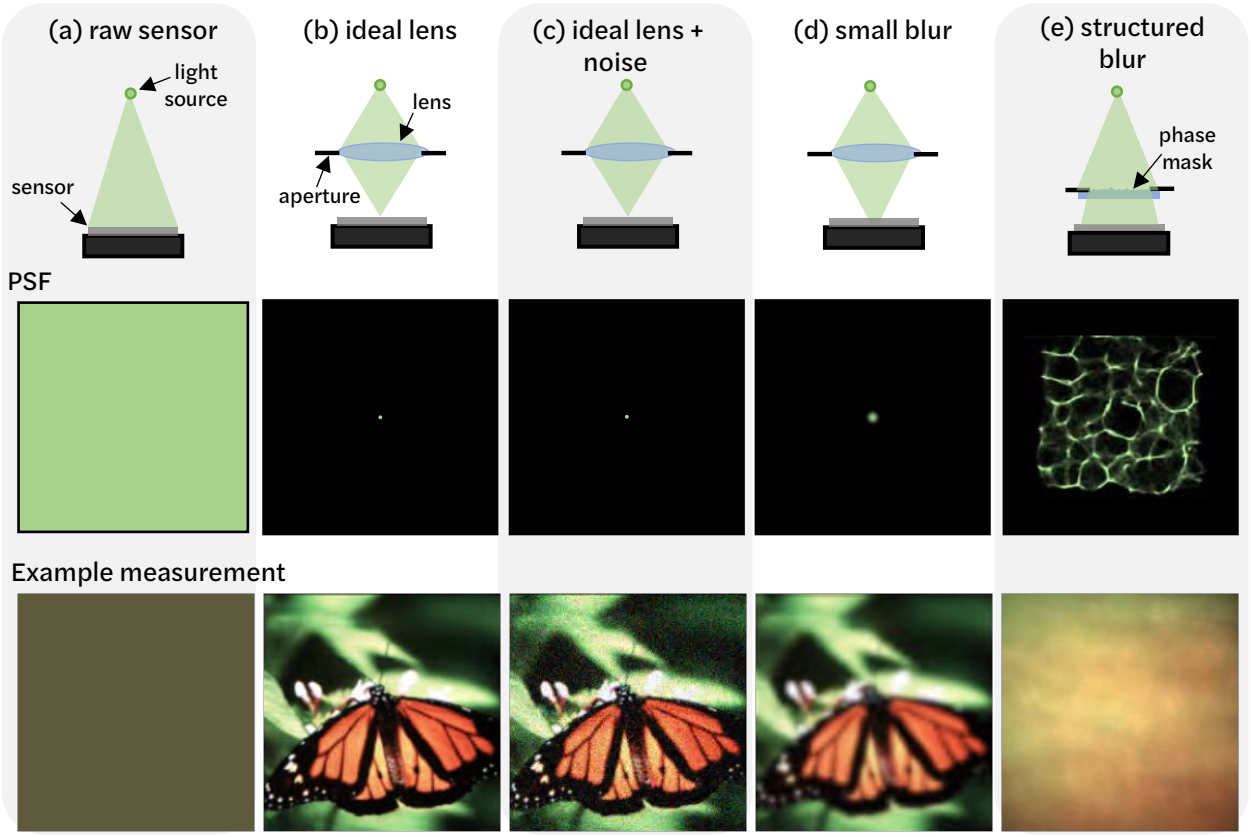


Figure 2.1: **Example imaging systems**. (a) A camera sensor measures light hitting it. Without a lens, a bare camera sensor will measure the average light in the scene. (b) A lens can be used to focus light from the world onto the camera sensor. An ideal lens is free of aberrations and has a delta point spread function (PSF), resulting in a clean and sharp image that perfectly matches the world. (c) In low light conditions, the measurement will often be corrupted by noise. (d) Realistically, most lenses add some amount of blurring to the measurement, due to aberrations in the lens. (e) Structured, intentional blur can be intentionally introduced to the camera, resulting in a measurement that does not look like the original object, but enables reconstruction of the original object via a computational reconstruction algorithm.

very common problem in low-light photography and microscopy, where there is very little light present in the scene. Removing the noise to extract the underlying signal in the image is the well-studied problem of image denoising [32, 161], and is the focus of Chapter 7.

For a non-ideal camera, we must model the effect of the optics of the camera on the image. Aberrations, or non-idealities, in the optics can lead to blurring at the image plane [87], Fig. 2.1(d). This blur can be unintentional, resulting from imperfections in the optics [83, 186], or camera motion [47], or it can be intentionally designed to encode additional information [8], Fig. 2.1(e). Computational cameras with intentionally designed blur will be

further discussed in Chapter 3, and the algorithms for these cameras will be elaborated on in Chapters 4, 5, 6.

To model how light passes through a lens with aberrations or blur, we can measure the impulse response function of the camera. To do this, we shine a point source, such as a flash light placed far away from the camera, through our imaging system, and measure the response on the camera. This is called the point spread function (PSF) of the camera. For an ideal lens with no aberrations, the PSF will be a delta function, whereas for a real lens with slight aberrations, the PSF will appear as a blurry point, Fig. 2.1(d). Systems with intentionally-designed blur may have much larger PSFs that span the entire size of the sensor and contain many sharp features, Fig. 2.1(e).

If one moves the point source around and the PSF remains constant, we can describe the PSF as being shift-invariant, Fig. 2.2(a). When the PSF is shift-invariant, each shift in object space results in a linear shift of the PSF. This provides us with a convenient mathematical model for our system, which can be modeled as a convolution:

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{h} * \mathbf{x}, \tag{2.3}$$

where $\mathbf{h}$ is the PSF of the camera. If one moves the point source around and the PSF varies with object lateral position, then the PSF is spatially-varying, Fig. 2.2(b). This leads to a more complicated calibration process, since the PSF must be identified at each field point, and to a more complicated forward model, since the convolutional model no longer applies. Under certain assumptions, other forward models, such as revolution-invariance [107], local convolutions [115], or low-rank approximations [223] can be used to simplify the calibration and forward model for such systems.

Other kinds of computational imaging systems, such as magnetic resonance imaging (MRI), computed tomography (CT), and phase retrieval, have distinct forward models that can be modeled in a similar way, producing a $\mathbf{y} = \mathbf{A}\mathbf{x}$ problem. Many methods and ideas described here are applicable and can be transferred to those other domains.

## 2.2 Inverse Problem: Recovering the Object from the Measurement

Given a measurement, $\mathbf{y}$, and system matrix, $\mathbf{A}$, we would like to recover the unknown object, $\mathbf{x}$. Naturally, the naive approach to solving this sort of inverse problem would be to attempt to invert the system matrix, $\mathbf{A}$, and to find:

$$\hat{\mathbf{x}} = \mathbf{A}^{-1}\mathbf{y} \tag{2.4}$$

Unfortunately, this often does not work very well for imaging inverse problems due to the size of the $\mathbf{A}$ matrix. For example, for a 12 megapixel camera, the size of $\mathbf{A}$ would be $(12 \times 10^6) \times (12 \times 10^6)$, since the image must be vectorized. Assuming the matrix is stored with each entry represented by an 8-bit integer value, this would result in 18 terabytes (TB) just to store the $\mathbf{A}$ matrix. Such a matrix inversion would be extremely computational costly

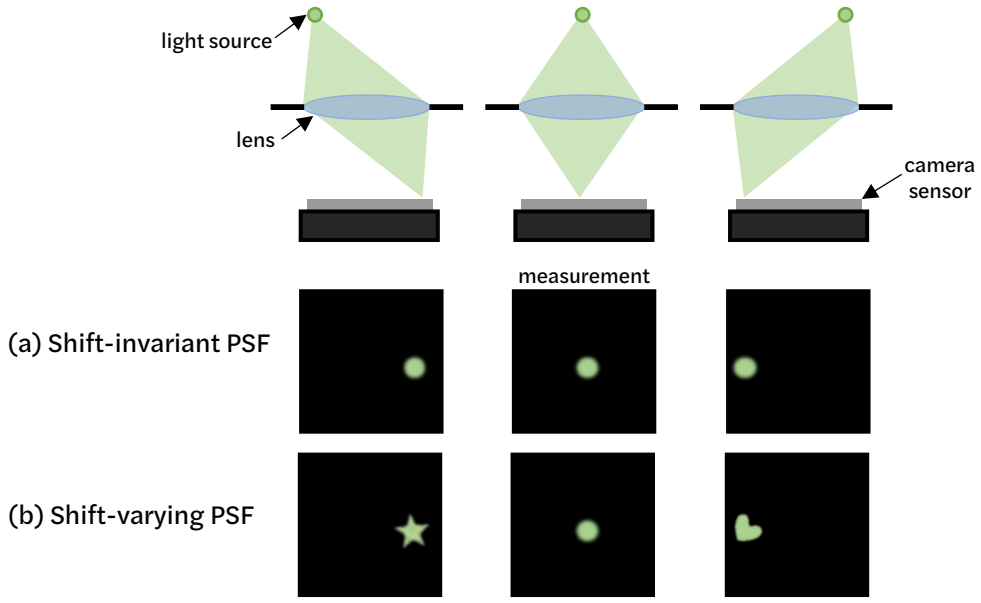


Figure 2.2: **Shift-invariant vs. spatially-varying imaging systems**. (a) In a shift-invariant system, the PSF remains the same for each shift of the point source. This results in a simple calibration process and allows a convolutional forward model to be used. (b) In a spatially-varying system, the PSF changes for different shifts of the point source, complicating the calibration and mathematical model.

and difficult to run on most computers. Due to this, we often do not explicitly calculate the $\mathbf{A}$ matrix, but rather implicitly define the operations associated with the $\mathbf{A}$ matrix.

Furthermore, we often aim to extract additional information for our encoded measurements, such as 3D from an encoded 2D measurement, Fig. 2.3. In this setting, we must solve an underdetermined system in which the number of unknowns is greater than the number of measurements. For underdetermined systems, the $\mathbf{A}$ matrix is non-square and a simple matrix inverse cannot be performed. Since there are infinite possible solutions for an underdetermined system, additional constraints are needed.

To solve inverse problems for imaging systems with large $\mathbf{A}$ matrices, that may be poorly conditioned or underdetermined, we often formulate an optimization problem that consists of a data fidelity term and a prior term,

$$\hat{\mathbf{x}} = \arg\min \mathcal{D}(\mathbf{x}; \mathbf{y}) + \tau \mathcal{P}(\mathbf{x}) \tag{2.5}$$

Here $D(\mathbf{x}; \mathbf{y})$ is the data fidelity term which makes sure our object guess is consistent with our measurement and our model of the system. The prior term, $\mathcal{P}(\mathbf{x})$ is hand-picked for a given application and adds additional constraints. For the data fidelity term, it is common to minimize the least square error between the measurement and the estimated measurement given the forward model and object guess,

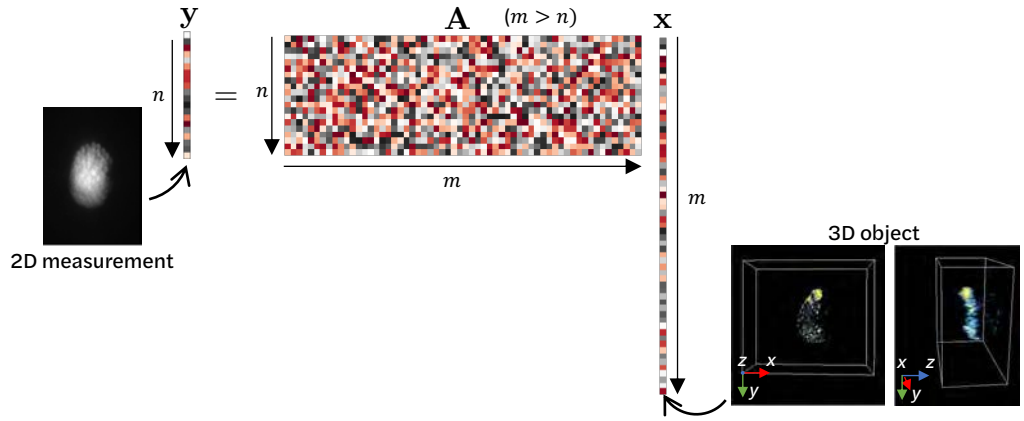


Figure 2.3: **Underdetermined imaging systems**. In an underdetermined imaging system, we aim to solve for more unknowns than measurements, such as recovering a 3D object from a 2D measurement. In this setting, our $\mathbf{A}$ matrix is wide, with more columns than rows and there are infinite possible solutions so additional constrains are needed for proper image recovery.

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\geq 0} \underbrace{\frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2}_{\text{data fidelity}} + \underbrace{\tau P(\mathbf{x})}_{\text{prior}} \tag{2.6}$$

There are many possible choices for priors. When $\mathbf{A}$ is ill-conditioned, but not underdetermined, adding a quadratic penalty term is common (Tikhonov regularization [201]), $\mathcal{P}(\mathbf{x}) = \|\mathbf{x}\|_2^2$. For underdetermined problems, an $l_1$ prior which promotes sparsity is a common choice. According to compressive sensing theory, it is possible to find the exact solution to an underdetermined system given two conditions are met: 1) signal sparsity in some domain (e.g. an image of the night sky is sparse in the native basis, since it consists of mostly zeros and a few non-zeros), and 2) incoherence in the measurement basis (e.g. a random sensing matrix) [37, 35]. By compressive sensing theory, an underdetermined imaging inverse problem that satisfies these conditions can be exactly recovered by minimizing,

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\geq 0} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \tau\|S\mathbf{x}\|_1 \tag{2.7}$$

Where $S$ is some sparsifying transform. For native sparsity, $S = I$. Other common sparsity bases include wavelets and discrete cosine transforms. A common prior for imaging inverse problems is total-variation (TV) regularization. This can be formulated as a penalty on the image gradients, promoting sparse image gradients: $\mathcal{P}(\mathbf{x}) = \|\nabla_{xy}\mathbf{x}\|_1$.

This general strategy of formulating an optimization problem with system knowledge and hand-picked priors has been used for many years to solve imaging inverse problems. We will refer to this general strategy as *classic methods*. More recently, deep learning has also been introduced for solving imaging inverse problems. In this setting, system knowledge and a

hand-picked prior are replaced with a trainable network and lots of training data. A trainable deep network serves as a function approximator, which aims to decode the measurement and directly solve the inverse problem. This requires the network to learn how to 'undo' the imaging system physics *and* learn a good imaging prior from the training data. We will refer to these approaches as *deep methods*. In this dissertation, we will introduce a hybrid approach called *physics-informed machine learning* in which system knowledge is built into the deep network, forming physics-informed network layers. Here, the network does not need to learn all of the imaging physics from scratch, and can instead use the training phase for further forward model refinement and to learn a good imaging prior. Figure 2.4 outlines the differences between these three methods.

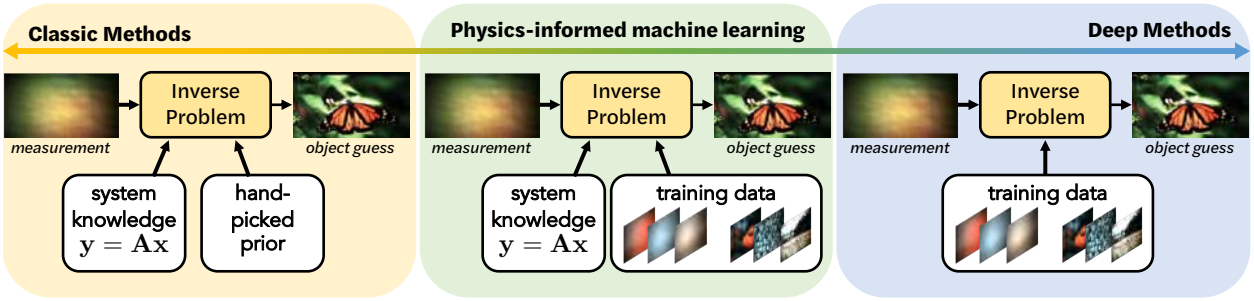


Figure 2.4: **Classic vs. deep vs. physics-informed methods**. Classic methods incorporate system knowledge and hand-picked priors to solve imaging inverse problems, but utilize no training data. Deep methods use neural networks and training data, but do not incorporate system knowledge. Physics-informed machine learning is a hybrid approach which incorporates system knowledge and neural networks through physics-informed layers. By including both system knowledge and training data to refine and improve the result, physics-informed networks can provide better results, require less training data, and maintain more interpretability than deep methods.

## Classic Methods

With classical methods, we aim to solve the imaging inverse problem by minimizing a hand-designed optimization problem that generally consists of a data fidelity term and a prior term, Eq. 2.6. There are many methods that have been developed over the years for solving these sorts of problems, such as Richardson Lucy [65], iterative shrinkage-thresholding algorithm (ISTA) [52], fast iterative shrinkage-thresholding algorithm (FISTA) [20], alternating direction method of multipliers (ADMM) [28], among others. Here, we will focus on two common methods, FISTA and ADMM, which are both types of proximal algorithms and generally perform well for large-scale imaging problems.

Proximal algorithms [166] are useful for solving optimization problems that have the form:

$$F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}), \tag{2.8}$$

where $f(\mathbf{x})$ is smooth and convex, but $g(\mathbf{x})$ is convex and non-smooth. In our case, the $l_1$ term promoting sparsity is non-smooth, resulting in an optimization problem that is convex, but not differentiable. Subgradient methods can be used to solve this problem, but they generally have poor convergence rates, $\mathcal{O}(1/\sqrt{k})$. Proximal algorithms, which repeatedly apply proximal operators, provide better convergence rates. Many proximal operators have closed form solutions, which are easy to compute. For example, for the $l_1$ norm, the proximal operator is soft-thresholding, $\text{prox}_g(u_i) = \text{soft}(u_i, \lambda) = \text{sign}(u_i)\max(|u_i| - \lambda, 0)$, which is easy to compute.

One of the simplest proximal algorithms for imaging inverse problems is ISTA. ISTA alternates between taking a gradient step with respect to $f(\mathbf{x})$, and taking a proximal step to enforce $g(\mathbf{x})$ [219].

**Algorithm 1** ISTA - proximal gradient method

**Problem class:** $\min_x F(x) = f(x) + g(x)$
$f, g : \mathbb{R}^n \to \mathbb{R}$ are convex, $\nabla f$ L-Lipschitz and $g$ non-smooth
**Initialize:** $\mathbf{x}_0 \in \mathbb{R}^n$
**Repeat:**
$\mathbf{w}_{k+1} \leftarrow \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)$ ▷ Note: here the step size is $1/L$
$\mathbf{x}_{k+1} \leftarrow \text{prox}_{g/L}(\mathbf{w}_k)$

When we plug in our imaging forward model, and the proximal operator for the $l_1$ norm, we obtain a simple algorithm that takes a gradient step, and then soft-thresholds the result based on a tuning parameter, $\lambda$, which tunes the sparsity. This algorithm has a slightly better convergence rate, $\mathcal{O}(1/k)$.

**Algorithm 2** ISTA - proximal gradient method for compressive imaging inverse problems

**Problem class:** $\min_x F(x) = f(x) + g(x)$
where $f(x) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ and $g(x) = \tau\|\mathbf{x}\|_1$
**Initialize:** $\mathbf{x}_0 = \mathbf{0}$
**Repeat:**
$\mathbf{w}_{k+1} \leftarrow \mathbf{x}_k - \frac{1}{L}\mathbf{A}^H(\mathbf{A}\mathbf{x}_k - \mathbf{y})$
$\mathbf{x}_{k+1} \leftarrow \text{soft}(\mathbf{w}_k, \lambda/L)$

Fast iterative shrinkage-thresholding algorithm (FISTA) can be used to obtain an even better convergence rate, $O(1/k^2)$, by taking a slightly more clever gradient step, Alg. 3. FISTA is easy to implement, and works well off-the-shelf given access to a known, well-calibrated forward model, $\mathbf{A}$. Here the sparsity parameter, $\lambda$, can be hand-tuned to obtain the best performance. In the past, using TV with FISTA required a nested optimization loop, however recent work has shown that it is possible to approximate TV in a single step with good precision, which is much faster [100].

**Algorithm 3** FISTA - accelerated proximal gradient

**Problem class:** $\min_x F(x) = f(x) + g(x)$
where $f(x) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ and $g(x) = \tau\|\mathbf{x}\|_1$
**Initialize:** $\mathbf{x}_0 \in \mathbb{R}^n, \mathbf{p}_1 \leftarrow \mathbf{x}_0,$ and $t_1 \leftarrow 1$
**Repeat:**
$\mathbf{w}_{k+1} \leftarrow \mathbf{p}_k - \frac{1}{L}\mathbf{A}^H(\mathbf{A}\mathbf{p}_k - \mathbf{y})$
$\mathbf{x}_{k+1} \leftarrow \text{soft}(\mathbf{w}_{k+1}, \lambda/L)$
$t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}, \beta_k \leftarrow \frac{t_k - 1}{t_{k+1}}$
$\mathbf{p}_{k+1} \leftarrow \mathbf{x}_k + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1})$

We note that for this type of optimization problem and many others, we do not need to explicitly generate the $\mathbf{A}$ matrix, and can instead implicitly model the operators associated with $\mathbf{A}$. For example, given a convolutional model, where $\mathbf{A}$ is a convolutional matrix associated with the operation $\mathbf{h} * \mathbf{x}$, we can represent $\mathbf{A}$, and $\mathbf{A}^H$ as:

$$\mathbf{A}\mathbf{x} = \mathbf{h} * \mathbf{x} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{h}) \cdot \mathcal{F}(\mathbf{x})) \tag{2.9}$$

$$\mathbf{A}^H\mathbf{x} = \mathbf{h} * \mathbf{x} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{h}^H) \cdot \mathcal{F}(\mathbf{x})) \tag{2.10}$$

where $\mathcal{F}$ is the Fourier transform, and $\mathcal{F}^{-1}$ is the inverse Fourier transform. Therefore, we only need to know our PSF, $\mathbf{h}$, and be able to calculate the hermitian of $\mathbf{h}$, which is simply the conjugate transpose of $\mathbf{h}$.

Another popular algorithm for imaging inverse problems is the alternating direction method of multipliers (ADMM) [28]. For ADMM, the objective terms are handled completely separately, so it is more amenable to handling a variety of constraints. ADMM is a popular choice for imaging inverse problems and for certain problems can speed up the reconstruction process [8].

This framework of setting up the inverse problem as an optimization problem is useful for a broad range of computational imaging problems with many different forward models. Generally, classic approaches work quite well off-the-shelf, but they have several issues. First, they rely on having a known forward model, $\mathbf{A}$. If the imaging model is unknown, approximated, or has noise/errors in it, then the solution will suffer. Next, these algorithms have several tuning parameters that control the sparsity, or the weights on different constraints. To find the best solution, a grid search on these parameters is often performed, which can be computationally costly. Next, these methods rely on hand-picked priors which may not be optimal for a given application, resulting in poor performance. Finally, these methods optimize a heuristic (consisting of the data fidelity term and the prior term), which may not result in the best image quality.

## Deep Methods

Recently, deep learning has been applied to copious different domains and achieved state-of-the-art performance [119, 127, 128, 163]. For an imaging inverse problem, a deep-learning based method attempts to solve the inverse problem through a deep network, which acts as a universal function approximator. The measurement is often an input to the network, and the network produces a guess of the object:

$$\hat{\mathbf{x}} = \mathcal{N}(\mathbf{y}) \tag{2.11}$$

For supervised machine learning, we must train our network before we can use it to solve imaging inverse problems, Fig. 2.5. To train the network, we need a large dataset of measurements and corresponding ground truth labels. It generally works best if this dataset comes from real data from the camera; however, in practice in can be hard or impossible to acquire aligned ground truth data for computational cameras. Alternatively, such a dataset can be simulated, but requires a very accurate forward model simulator for the camera to account for any effects that are present in the real system.

During training, we aim to solve the following optimization problem:

$$\arg\min_{\mathcal{N}} \sum_{i=0}^{N_t} \mathcal{L}(\mathbf{x}_i - \mathcal{N}(\mathbf{y}_i)), \tag{2.12}$$

where $\mathcal{L}$ is the loss function which calculates a distance between the network output, $\hat{\mathbf{x}}_i = \mathcal{N}(\mathbf{y}_i)$, and the ground truth label, $\mathbf{x}_i$. Some common loss functions include mean-squared error, $l_1$ loss, and perceptual losses such as Learned Perceptual Image Patch Similarity (LPIPS), which outputs a perceptual metric [231]. Typically, the smaller the loss, the closer the two images are to each other. During training, backpropagation is used to update the network weights based on the gradient of the loss function with respect to the network weights. The training phase lasts for many epochs, in which we cycle through all of the images in our training dataset in a random order. After training is complete, the network weights are held constant and the network should be able to solve the inverse problem for new measurements that it did not see during training.

While the training phase can be very lengthy, taking hours or days for certain problems, the testing phase can be very fast. One advantage of deep methods over classical methods is that excluding training time, deep methods can solve the inverse problem much faster than classic methods. While classic methods often solve an iterative optimization problem that takes many iterations to converge, deep methods require only a single forward pass through the network to solve the inverse problem, often resulting in order of magnitude speed ups over classic methods.

Furthermore, deep methods may be able to provide better image quality than classic methods. This is due to the fact that the training objective can be set to directly minimize the distance to the ground truth image, optimizing for image quality rather than some heuristic based on the data fidelity term and a hand-picked prior. In addition, the deep

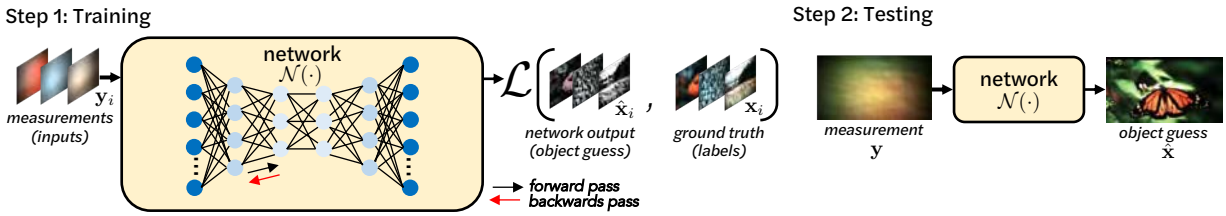


Figure 2.5: **Deep-learning-based reconstructions**. Deep-learning-based methods consist of two phases: training and testing. During training, a large dataset of labeled image pairs are used to update the network weights based on the loss between the ground truth label and the network output. After training is complete, the network is fed new data that it did not see during training, and if all goes well, the network should generalize and be able to solve the imaging inverse problem for new scenes.

network may be able to learn a better prior based on the training data than an arbitrary hand-picked prior.

On the other hand, deep methods are black-boxes and are limited by their datasets. There are few guarantees for when a deep network will be able to accurately solve an imaging inverse problem, and when it will hallucinate information. If the dataset is too small, or too restrictive, the network may not generalize well to other types of data.

In addition, the training phase usually needs to be repeated for each different camera. Acquiring a real ground truth dataset of image pairs can be costly and difficult, and this process needs to be repeated for each camera, adding an expensive calibration step to the computational imaging pipeline.

## Physics-informed machine learning

In this dissertation, we focus on a middle ground approach which we call *physics-informed machine learning*. We combine the best of classic and deep methods in order to solve our imaging inverse problems. To leverage our knowledge of imaging system physics, we use differentiable optical models. There are many ways of doing this, including algorithm unrolling [160, 228, 164], single-step physics-based inversion, and differentiable forward models. We will separate the methods into three different categories: supervised learning for inverse problems, unsuperivsed learning for inverse problems, and learning forward models.

### Supervised learning for inverse problems

In the case of supervised learning for inverse problems, we assume that we have access to a dataset of labeled data, consisting of measurements and their corresponding ground truth image pairs. This dataset can be experimentally captured, simulated, or perhaps generated. In addition, we assume that we have some knowledge of the imaging system physics—perhaps we have calibrated our imaging system and know our forward model, $\mathbf{A}$, or we know an

approximate version of our forward model, $\tilde{\mathbf{A}}$. Now, we aim to infuse some of this known physics-based knowledge into our neural network, so that we do not have to learn what we know or can model from scratch. In general, there are two common approaches to infusing this knowledge into our neural network: approximate physics-based inversion, Fig. 2.6(a), or algorithm unrolling Fig. 2.6(b).

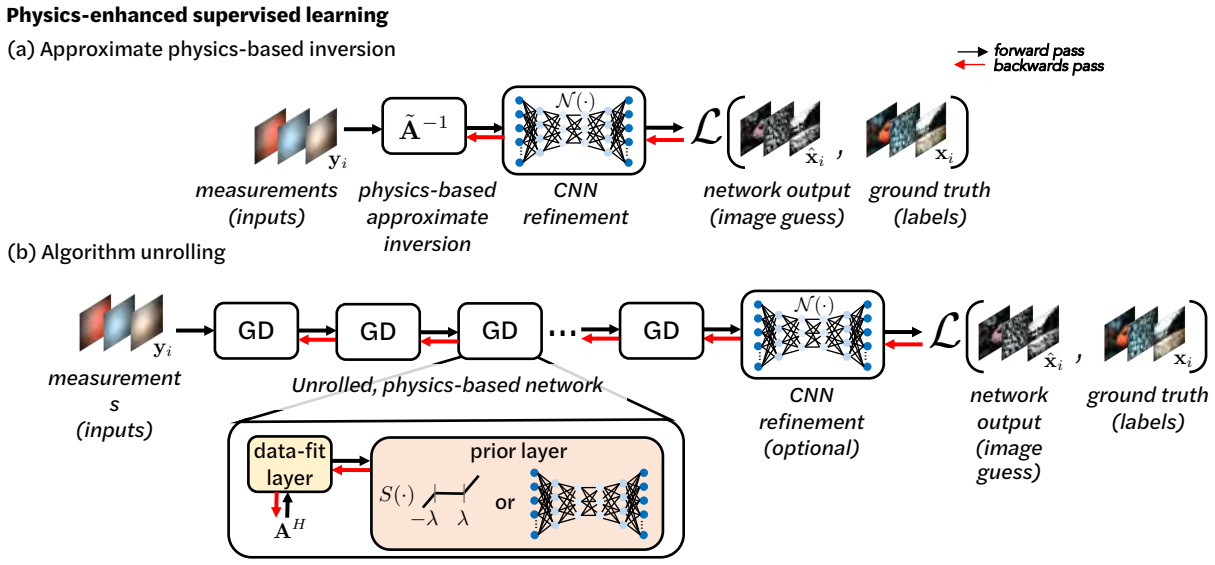


Figure 2.6: **Physics-informed supervised learning**. For supervised learning in which we have access to a dataset of paired measurements and labels, two common methods for structuring physics-informed learning include (a) approximate physics-based inversion followed by refinement with a neural network and (b) algorithm unrolling.

For approximate physics-based inversion, we aim to first approximately invert the effect of our optics, and then refine and improve our guess based on a neural network [105, 164]. Ideally, we could take the inverse of $\mathbf{A}$, however this is often not possible as described earlier. Rather than computing the inverse, the adjoint, $\mathbf{A}^H$, is often more feasible to compute and can be useful in 'unscrambling' some of the information. Alternatively, other single-step inversion methods, such as Wiener filters can be used in this step. Through differentiable modeling, the physics-based inversion layer can also be parameterized and refined together with the neural network. This is useful for when the optical model is not fully known or is not perfect. This method of a physics-based approximate inversion followed by a convolutional neural network (CNN) [81] for refinement is simple to implement and generally works quite well in practice. We will demonstrate an example of this approach in Ch 5 for the case of single-shot 3D microscopy with spatially-varying aberrations, showing how we can use an approximate physics-based inversion based on an easier-to-model spatially-invariant approximation and then refine this approximation through our CNN.

Another approach is to use algorithm unrolling to create physics-inspired layers. In this approach, we can take an optimization algorithm, such as FISTA or ADMM, and 'unroll'

it [80, 184, 56, 228, 193]. For example, for ISTA, rather than applying the two update steps in Alg. 2 over and over again until convergence, we can unroll the algorithm and apply those steps $N$-times, where $N$ is small. Figure 2.6(b) demonstrates this concept. For our ISTA example, the data-fit layer corresponds to the update-step: $\mathbf{w}_{k+1} \leftarrow \mathbf{x}_k - \frac{1}{L}\mathbf{A}^H(\mathbf{A}\mathbf{x}_k - \mathbf{y})$, and the prior layer corresponds to the update-step: $\mathbf{x}_{k+1} \leftarrow \text{soft}(\mathbf{w}_k, \lambda/L)$. These update steps can be written in a differentiable way, and the parameters of these unrolled layers can then be optimized. In this simplest case, the parameter $\lambda$, which is a tuning parameter controlling sparsity, can be updated based on the dataset to achieve images that are the most similar to the ground truth based on some loss metric. This parameter could be globally set across the whole network, or optimized per layer. Alternatively, more parameters, such as the sparsifying transform and the proximal function can be learned to achieve the best performance, or the entire prior term can be replaced with a neural network, alternating between a data-fit term and a neural network which enforces some prior on the scene statistics.

When the forward model is unknown, or imperfect, $\mathbf{A}$ could also be parameterized and updated to achieve the best reconstruction quality. In addition, a neural network can be optionally added after the unrolled network and serve as an additional refinement step to further improve image quality. Unrolled, physics-based networks are slightly more complicated to implement than single-step inversions, but can achieve good performance. Compared to classic methods, unrolled networks can be much faster, since we can bound the number of layers and learn the best parameters for each layer, achieving comparable performance to a fully converged algorithm in a fraction of the time. In their most basic form unrolled algorithms with optionally learnable hyperparameters (e.g. $\lambda$) are interpretable and fairly reliable. As the number of layers increases, memory also increases, making this network potentially very memory-intensive. Techniques such as check-pointing and memory-efficient learning can mitigate this [102]. In addition, recent work on implicit differentiation could allow us to fully compute until convergence an optimization algorithm such as FISTA, while still keeping certain parameters differentiable [23]. Unrolled optimization and implicit differentiation show promise for interpretable end-to-end learning of optics, in which we aim to learn the best optics given a reconstruction algorithm [103, 13]. We will demonstrate an example of an unrolled, physics-based network based on ADMM in Ch. 4 for lensless imaging reconstructions.

### Unsupervised learning for inverse problems

Unsupervised learning is a promising area of research because it does not require a dataset of measurements and corresponding ground truth images. Recently, a technique called Deep Image Prior [204] showed remarkable results for a variety of tasks, such as super-resolution, denoising, and image in-painting, using an untrained network. This demonstrates that the structure of a neural network can function as a prior on the image statistics, even if the network is not trained. For physics-informed unsupervised learning, we use the untrained network as an image generator, constraining the domain of possible images to be the output of our generator, $\hat{\mathbf{x}} = \mathbf{G}(z; W)$. We then feed the generated image through a differentiable

optical forward model, $\mathbf{A}$, to generate a measurement guess, $\hat{\mathbf{y}} = \mathbf{A}\hat{\mathbf{x}}$, and compare our real measurement, $\mathbf{y}$, with our generated measurement. Using the loss between these two values, we update the weights of the network through backpropagation until the measurement and our generated measurement match, Fig. 2.7. This can be described as solving the following optimization problem:

$$\arg\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{G}(z; W)\|_2^2. \tag{2.13}$$

When comparing this equation to Eq. 2.6, we can see that this is essentially optimizing the data-consistency term. Rather than having an explicit prior, the prior is obtained through the structure of the network, $\mathbf{G}(z; W)$. Unsupervised learning has been shown to be useful for a number of imaging inverse problems, such as MRI [76, 49, 98] and diffraction tomography [236]. In addition, when $\mathbf{A}$ is partially known and parameterizable, this strategy has shown success in jointly reconstructing *and* calibrating the forward model, $\mathbf{A}$ at the same time [25].

Overall, unsupervised learning often has better image quality than classic methods, but may have worse image quality than supervised approaches. In addition, the network must often be retrained for each new measurement, which can be quite slow—slower than both supervised methods and classic methods. Despite these disadvantages, the advantage of not needing paired training data is a big one, and makes this class of reconstruction methods very promising for a number of applications. We will demonstrate an example of an untrained network for compressive hyperspectral imaging and video from stills in Ch. 6.

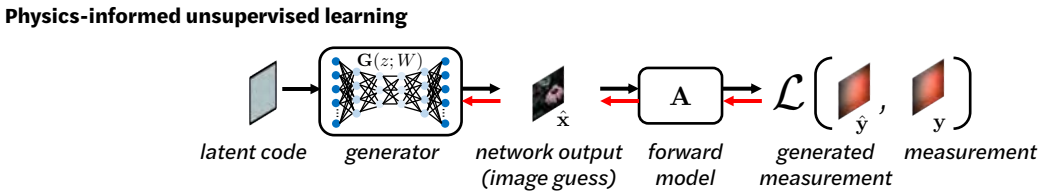


Figure 2.7: **Physics-informed unsupervised learning**. For physics-informed unsupervised learning, the neural network structure can serve as an imaging prior. A data-consistency loss (the difference between the real measurement and the guess of the measurement based on our forward model) is computed, then used to update the value of the network weights.

### Learning forward models

Rather than using a neural network to solve an imaging inverse problem, it is sometimes useful to use a neural network to help us learn, calibrate, or characterize an imaging forward model, $\mathbf{A}$. In this way, we can use a neural network to represent a complicated, hard to compute, or perhaps non-linear imaging forward model. As a simple example, we can use

data to calibrate a known imaging forward model. For instance, if we can easily parameterize our imaging forward model, but do not know the values of the parameters (e.g. unknown Zernike coefficients for aberrations), we could express our optical model in a differentiable way, then use backpropagation to update our parameters based on the loss between the real measurements and generated measurements from our optical model (given a dataset of measurements and ground truth labels). For a more complicated system which we cannot easily parameterize or model, it may be useful to parameterize and model as much as we can, then combine this with a neural network to learn the rest, Fig. 2.8.

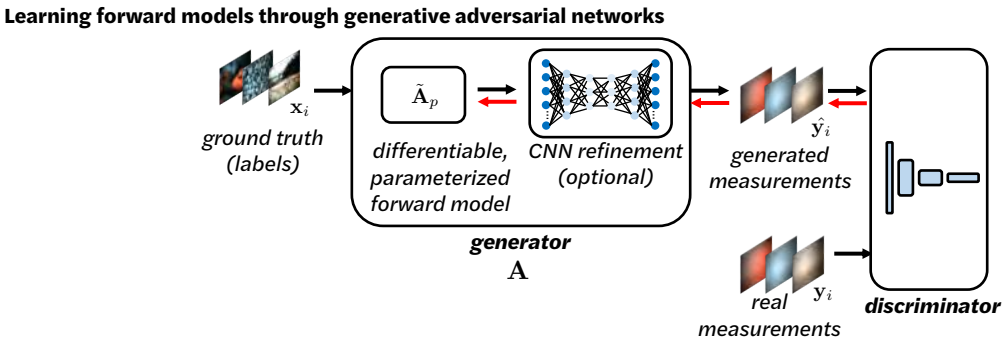


Figure 2.8: **Learning imaging forward models**. We can use a generative adversarial network (GAN) to help us calibrate imaging forward models.

For cases where there is lots of noise in the measurement, stochastic components within the forward model (e.g. modeling noise), or a lack of paired training data, a generative adversarial network (GAN) [77] can be used. This is useful when it is hard to compute a pixel-wise loss between the measurements, $\mathbf{y}_i$, and generated measurements, $\hat{\mathbf{y}}_i$, (e.g. in the presence of noise). Rather than directly computing the loss between $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$, we can instead feed these two images into a discriminator. The discriminator is another neural network whose job is to decide whether a measurement is real or fake. The generator aims to create plausible measurements that match the distribution of the real measurements, while the discriminator aims to classify whether the measurements are real or not. These two networks are trained together and compete with each other.

The generated measurement is: $\hat{\mathbf{y}}_i = G(\mathbf{x}_i)$. When updating the parameters of the discriminator, we can calculate a loss based on how well the discriminator classifies the generated measurements, $\hat{\mathbf{y}}_i$, as being fake and the real measurements, $\mathbf{y}_i$, as being real. When updating the parameters of the generator, we calculate a loss based on how well the generator can fool the discriminator and get the discriminator to classify the generated measurements as being real. At the end of training, the generator should be able to fool the discriminator 50% of the time and the distribution of the generated measurements should be indistinguishable from the real measurements.

In practice, GANs are notoriously difficult to train. Over the years, a number of methods, such as Wasserstein GANs and gradient penalties have been introduced to improve the

stability of GANs and make GANs easier to train and debug [82, 11]. Constraining the generator in a GAN to learn only a few physics-inspired parameters rather than millions of parameters in a CNN is a promising direction for making GANs easier to train and more interpretable. We will demonstrate an example of using a physics-informed GAN to learn a signal-dependant, camera-dependent extremely low light noise model in Ch. 7.

# Chapter 3

# Compressive Lensless Imaging: Single-shot Hyperspectral, 3D, and Video

In this chapter, we introduce compressive, mask-based lensless cameras based on DiffuserCam, a compact and inexpensive single-shot 3D camera [8]. First, we provide intuition for how to encode additional information (hyperspectral, video, 3D) using a lensless camera and examine the forward model for each case. Next, we summarize the classic inverse problem for these cameras, discussing compressive sensing and the need for sparsity as well as good priors. Finally, as a case study of lensless imagers, we take a deep dive into the design of Spectral DiffuserCam[1], which is a snapshot hyperspectral imager. We provide context for this camera in the field of compressive hyperspectral imagers, provide design details, discuss limitations, and showcase the performance of this camera when using classic reconstruction methods.

## 3.1 Modeling Mask-based Lensless Cameras

Mask-based lensless imagers, often called lensless cameras, are a class of computational cameras in which the lens is replaced with a phase or amplitude mask placed a short distance in front of the sensor [8, 15]. Unlike conventional (lensed) cameras, which directly record a one-to-one image of the scene, lensless cameras map each point in the scene to many sensor pixels, *indirectly* encoding scene information into the sensor measurement. A reconstruction algorithm is then used to recover the final image. This architecture enables small, cheap, and light-weight designs which can be used for portable or *in vivo* imaging [14, 199, 8, 112, 222, 132, 113, 198]. Additionally, the inherent multiplexing of lensless cameras can make them amenable to compressive measurement of higher-dimensional signals, such as

[1]This chapter is based on the published journal paper titled "Spectral DiffuserCam: lensless snapshot hyperspectral imaging with a spectral filter array" and is joint work with Kyrollos Yanny, Neerja Aggarwal, and Laura Waller [158].

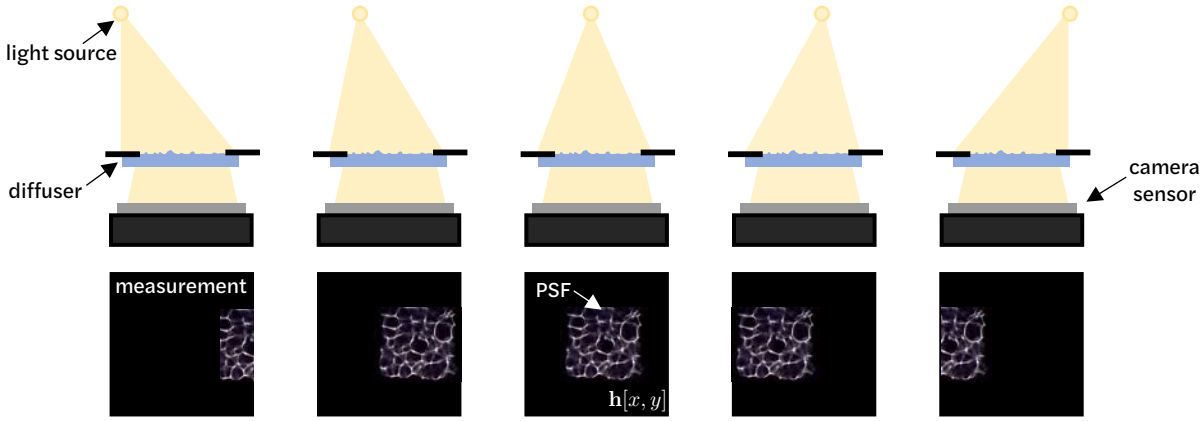


Figure 3.1: **DiffuserCam forward model**. DiffuserCam is a mask-based, lensless camera that consists of a thin diffuser placed close to the camera sensor. A point source in the world maps to a high contrast, wavy pattern on the sensor (PSF). As the point source moves laterally through the world, the PSF linearly shifts across the sensor (shift-invariant).

3D volumetric [8, 91], hyperspectral [158], polarization [62], or video [9], from a single 2D measurement.

To demonstrate the modeling and algorithms for mask-based lensless cameras, we will focus our discussion on DiffuserCam [8, 113], which is a lensless camera based on a phase mask, or diffuser. DiffuserCam is a compact, easy-to-build imaging system that consists only of a diffuser (a transparent phase mask with pseudo-random, slowly-varying thickness) placed a few millimeters in front of a standard image sensor (see Fig. 3.1). First, we will develop a model for 2D imaging with DiffuserCam, then extend this to single-shot hyperspectral, video, and 3D imaging.

## 2D lensless imaging

Light from a point source in the scene is refracted by the diffuser to create a high-contrast caustic pattern on the sensor, which is the point spread function (PSF) of the system. Since the diffuser is thin and the sensor is placed within the caustic plane of the diffuser [8], the PSF can be modeled as shift-invariant: a lateral shift of the point source in the scene causes a translation of the PSF in the opposite direction, Fig. 3.1. We can therefore model the sensor response $\mathbf{b}[x, y]$ as a convolution of the scene $\mathbf{x}[x, y]$ with an on-axis PSF $\mathbf{h}[x, y]$:

$$\mathbf{b}[x, y] = \text{crop}\Big(\mathbf{x}[x, y] * \mathbf{h}[x, y])\Big) \tag{3.1}$$

$$= \mathbf{A}_{\text{2D}}\mathbf{x}, \tag{3.2}$$

where $[x, y]$ represents the discrete image coordinates, $*$ represents a discrete 2D linear convolution and the crop function accounts for the finite sensor size. In practice, it is

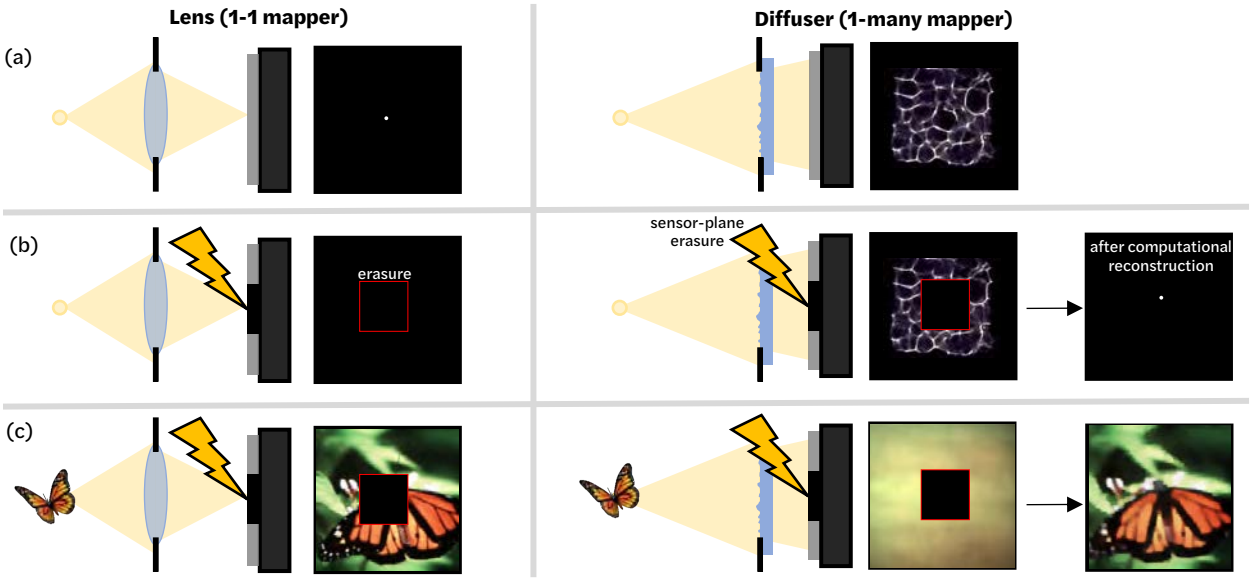


Figure 3.2: **1-1 imagers vs. 1-many imagers**. (a) Comparing a 1-1 imager to a 1-many imager. In the presence of sensor-plane erasures (b), the information from a 1-1 imager may be lost. On the other hand, a 1-many imager is resilient to erasures, since information is spread across the sensor. After computational recovery, the object can be almost fully recovered. (c) This is true for simple objects, as well as more complicated scenes.

necessary to pad both the image and the PSF before performing the convolution since FFT-based convolutions have circular boundary conditions, causing the edges of the field of view to wrap around the other side. The crop function therefore brings the image and PSF sizes back to the original size after the convolution is performed. For 2D imaging, we assume that the PSF does not vary with depth—that is, as one moves objects closer or further from the sensor, the PSF will remain the same. This assumption is true as long as objects are placed beyond in the hyperfocal distance of the camera, which in the case of DiffuserCam can be between a few centimeters to a few meters away from the camera [8].

## 2D imaging with erasures

To understand how we can encode additional, higher-dimensional information with DiffuserCam, let us first consider a simpler problem: recovering information from a sensor that has erasures, or pixel defects. A typical lens-based camera is a 1-1 mapper, so each point in the world maps to a single point on the sensor, Fig. 3.2(a)(left). On the other hand, DiffuserCam is a 1-many mapper, so each point in the world maps to a random pattern on the sensor, which consists of many points across a large portion of the sensor, Fig. 3.2(a)(right). If we destroy a square of pixels in the center of the sensor, a lens-based camera will not record the light from the center of the field of view, Fig. 3.2(b)(left). When using a diffuser, light from the center of the field of view will hit many pixels across the sensor, so even if the

center pixels are destroyed, information from the center of the field of view is still measured, Fig. 3.2(b)(right). We can imagine that given a good reconstruction algorithm, we could recover the object from the incomplete, erased measurement in the case of DiffuserCam. In the case of the lens, the information is not measured, so it cannot be recovered and must instead be guessed. This concept can be extended to more complicated scenes, Fig. 3.2(c).

The forward model for a lensless camera with focal plane erasures is:

$$\mathbf{b}[x,y] = \mathbf{M}[x,y] \cdot \text{crop}\Big(\mathbf{x}[x,y] * \mathbf{h}[x,y])\Big) \tag{3.3}$$

$$= \mathbf{A}_{\text{2D erasures}}\mathbf{x}. \tag{3.4}$$

Where $\mathbf{M}[x,y]$ is a binary mask that zeros out a subset of the sensor pixels corresponding to the erasure pattern. Although commercial sensors are screened to minimize dead pixels, this scenario is useful to help us explore the limits of computational recovery with these sorts of cameras. How many pixels can be erased while maintaining decent object recovery? How does the reconstruction degrade as a function of pixel erasures? We will explore these questions in detail in Chapter 6.

Since not all of the sensor pixels are needed to fully or almost fully recover the object, one might ask: can we do something useful with the erased/unneeded pixels? This brings us to our next concept of snapshot hyperspectral and video imaging, in which we can utilize different subsets of pixels to accomplish different tasks. In the case of hyperspectral imaging, we can encode different wavelengths of light into different subsets of pixels, then fully recover a high-resolution object at each wavelength, Fig. 3.3(c). For the case of single-shot video, we can encode different points in time across different pixels, recovering a full video from a single image, Fig. 3.3(b).

## Higher dimensions: single-shot video and hyperspectral imaging

Since each point in the world maps to many pixels on the sensor, we need only a subset of the sensor pixels to reconstruct the full 2D image, given certain conditions such as sufficient sparsity. This can be useful, for example, to compensate for dead sensor pixels, which act like an erasure pattern; or, we can capture the full 2D sensor image and recover higher-dimensional information. The higher-dimensional data (e.g. time, wavelength) must be physically encoded into different subsets of the sensor pixels; then compressed sensing is applied to recover a 3D datacube.

For single-shot video, we can use the rolling shutter of the camera to encode temporal information into different rows of the camera, Fig. 3.3(b). Each row contains information from different time points, then using algorithms, we can recover full images from each subset of rows, forming a full, high-resolution video from a single image [9]. For hyperspectral imaging, we can instead encode different wavelengths of light into different regions on the sensor, then recover a full hyperspectral datacube from a single encoded image Fig. 3.3(c). This can be accomplished with the use of spectral filter arrays attached to the imaging

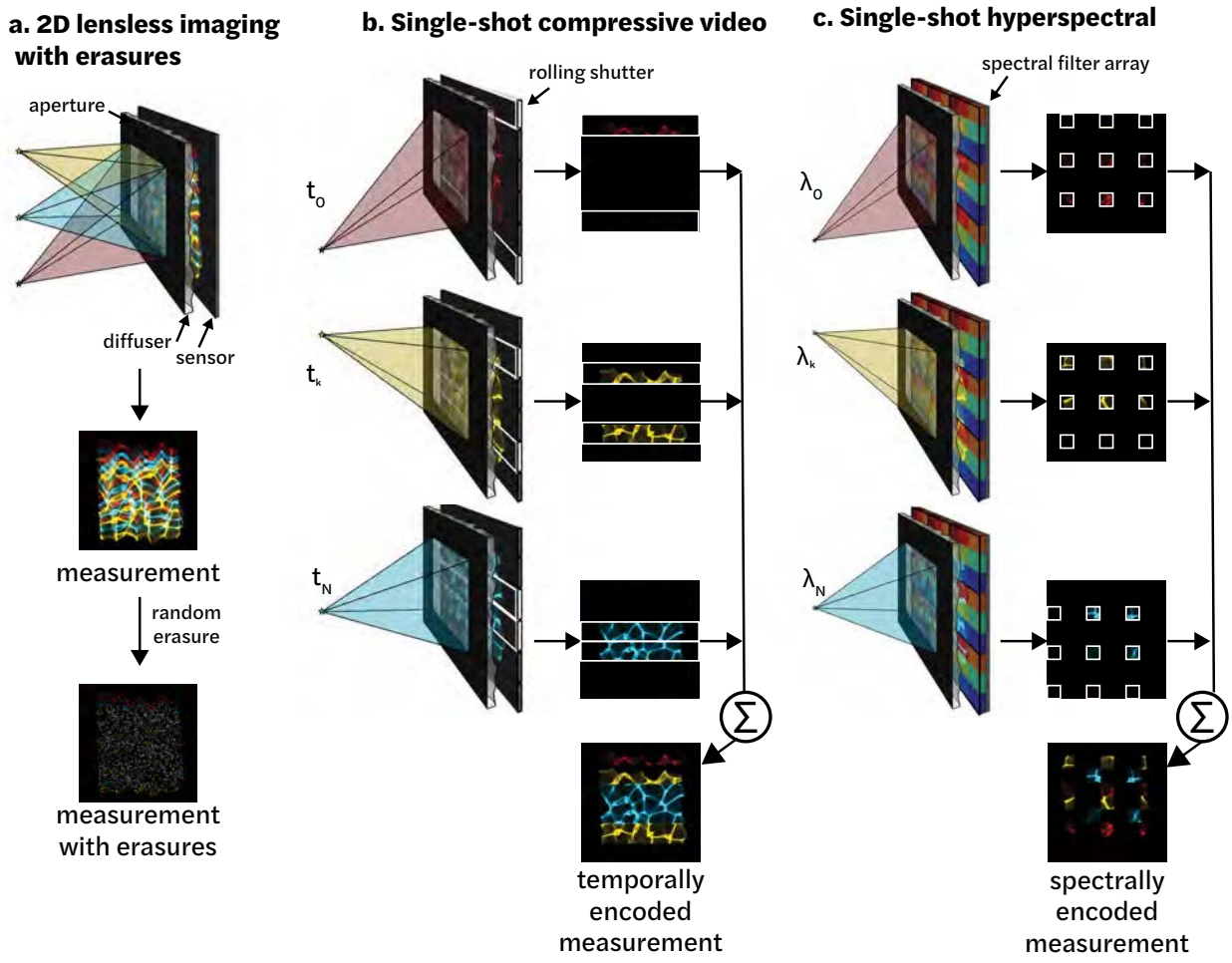


Figure 3.3: **Compressive imaging systems:** Our lensless 2D camera consists of a diffuser placed a small distance away from the sensor. Each point in the world maps to a high-contrast caustic pattern (PSF). (a) Since the PSF covers a large area of the sensor, we can erase a random subset of the pixels and still recover the full image. (b) The camera's rolling shutter can be used to encode temporal information into the measurement. As points flick on/off, the PSFs are filtered by the sensor's rolling shutter function, which reads one row at a time (or two in the case of dual shutter, as shown here). The measurement is a sum of the rows captured at different time points, each of which contains information from the entire 2D scene. Hence, temporal information is encoded vertically across the sensor, with earlier time points at the top and bottom of the image and later time points towards the center. (c) Single-shot hyperspectral imaging is achieved with a spectral filter array in front of the sensor, which maps each band of wavelengths to a subset of the sensor pixels.

sensor [158]. In this way, rather than sampling only a subset of pixels in order to reconstruct a 2D image, we instead can reconstruct 3D datacubes from fully-sampled 2D images.

To update our imaging model to account for higher-dimensions, we denote this extra dimension (time/wavelength) generically as the $k$-dimension. Sequential recovery using Eq. 3.3 prevents incorporating priors along the $k$-dimension, so we use the following model that depends on the full datacube:

$$\mathbf{b} = \sum_{k=0}^{N_k} \mathbf{M}_k[x,y] \cdot \text{crop}\Big(\mathbf{x}[x,y,k] * \mathbf{h}[x,y]\Big) \tag{3.5}$$

$$= \mathbf{A}_{k\text{-D}}\mathbf{x}. \tag{3.6}$$

Here, $N_k$ is the number of discrete points along the $k$-dimension, and $\mathbf{M}_k[x,y]$ is a masking function, which depends on $k$, and selects the sensor pixels corresponding to each video frame/wavelength. The convolution, $*$ is only over the two spatial dimensions.

For the high-speed video case, $\mathbf{M}_k[x,y]$, referred to as the shutter function, is based on the rolling shutter. Sensors typically have either a single shutter or a dual shutter, which we use in this work. For a single shutter, the sensor reads the pixels one horizontal line at a time, moving from the top to the bottom of the sensor; for a dual-shutter, the sensor reads pixels from two horizontal lines that move from the top and bottom of the sensor towards the middle, Fig. 3.3(b).

For the hyperspectral case $\mathbf{M}_k[x,y]$, referred to as the filter function, is determined by the spectral filter array. Each filter pixel acts as narrow-band spectral filter which integrates light within a certain wavelength range and blocks out light at other wavelengths. We approximate this as a finite sum across spectral bands with a non-binary filter function which accounts for the spectral transmittance at each wavelength.

## Higher dimensions: single-shot 3D

Rather than exploiting different regions of the sensor to encode higher-dimensional data, we can also exploit the behavior of the PSF to encode additional information. For example, if the PSF varies with depth [8, 223] or wavelength [75], this can be used to encode higher-dimensional information into the measurement.

For a system with a depth-varying PSF, we have the following forward model:

$$\mathbf{b} = \sum_{z=0}^{k} \text{crop}\Big(\mathbf{x}[x,y,z] * \mathbf{h}[x,y,z]\Big) \tag{3.7}$$

$$= \mathbf{A}_{3\text{-D}}\mathbf{x}. \tag{3.8}$$

where the PSF, $\mathbf{h}[x,y,z]$, now varies with depth, and we sum across $k$ discrete depth planes. To successfully encode and decode depth information, the PSF must sufficiently vary with depth. This can be measured by computing the correlation of PSFs at adjacent depths [8].

### Inverse problem for compressive imaging

Given our sensor measurement $\mathbf{b}$, our goal is to recover the scene $\mathbf{x}$. For the 2D erasures scenario $\mathbf{x}$ is a 2D image, whereas for single-shot video $\mathbf{x}$ is a video consisting of two spatial and one temporal dimension, for single-shot hyperspectral $\mathbf{x}$ is a hyperspectral volume consisting of two spatial and one spectral dimension, and for single-shot 3D $\mathbf{x}$ is a volume consisting of three spatial dimensions. In all cases, the problem is underdetermined, since we aim to recover more pixels than we measure. When using a classic method, we can solve the following optimization problem:

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \geq 0}{\arg\min} \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \tau \|\nabla_{xyk} \mathbf{x}\|_1.$$

We can choose the sparsifying transform (e.g. native sparsity, wavelets, TV) that gives us the best performance and then tune $\tau$ to trade off data-fidelity and our sparsity prior until we achieve the best results. By compressive sensing theory, given certain assumptions about scene sparsity in some domain and the incoherence of the imaging system, exact recovery of the high-dimensional object from the encoded measurement is possible [35]. Alternatively, this problem could be solved using deep learning or physics-informed learning if we have access to a dataset of image pairs. For compressive imaging, acquiring a dataset of image pairs can be tricky, since our ground truth data is 3D (hyperspectral, time, depth), while our measurements are 2D.

## 3.2 Spectral DiffuserCam: a Compact Camera for Snapshot Hyperspectral Imaging

Next, we go into depth on the design of a compressive camera, Spectral DiffuserCam, which we developed for single-shot hyperspectral imaging. This next section highlights the design choices for our camera, provides context for its place within the field of computational imaging, and highlights the performance that is possible using classic methods.

### Introduction

Hyperspectral imaging systems aim to capture a 3D spatio-spectral datacube containing spectral information for each spatial location. This enables the detection and classification of different material properties through spectral fingerprints, which cannot be seen with an RGB camera alone. Hyperspectral imaging has been shown to be useful for a variety of applications, from agricultural crop monitoring to medical diagnostics, microscopy, and food quality analysis [53, 104, 138, 196, 78, 5, 139, 165, 93, 17]. Despite the potential utility, commercial hyperspectral cameras range from \$25,000–\$100,000 (at the time of publication of this dissertation). This high price point and the large size have limited the widespread use of hyperspectral imagers.

Traditional hyperspectral imagers rely on scanning either the spectral or spatial dimension of the hyperspectral cube with spectral filters or line-scanning [79, 71, 227]. These methods can be slow and generally require precise moving parts, increasing the camera complexity. More recently, snapshot techniques have emerged, enabling capture of the full hyperspectral data cube in a single shot. Some snapshot methods trade-off spatial resolution for spectral resolution by using a color filter array or splitting up the camera's field-of-view (FOV). Computational imaging approaches can circumvent this trade-off by spatio-spectrally encoding the incoming light, then solving a compressive sensing inverse problem to recover the spectral cube [209], assuming some structure in the scene. These systems are typically table-top instruments with bulky relay lenses, prisms, or diffractive elements, suitable for laboratory experiments, but not the real world. Recently, several compact snapshot hyperspectral imagers have been demonstrated that encode spatio-spectral information with a single optic, enabling a practical form factor [180, 69, 95]. Using a single optic to control both the spectral and spatial resolution, they are generally constrained to measuring contiguous spectral bins within a given spectral band.

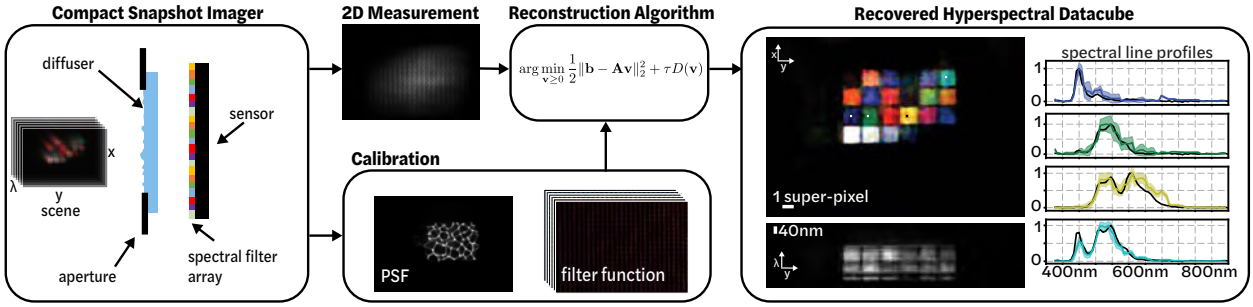


Figure 3.4: **Overview of the Spectral DiffuserCam imaging pipeline**, which reconstructs a hyperspectral datacube from a single-shot 2D measurement. The system consists of a diffuser and spectral filter array bonded to an image sensor. A one-time calibration procedure measures the point spread function (PSF) and filter function. Images are reconstructed using a non-linear inverse problem solver with a sparsity prior. The result is a 3D hyperspectral cube with 64 channels of spectral information for each of 448×320 spatial points, generated from a 2D sensor measurement that is 448×320 pixels.

Here, we propose a new encoding scheme that takes advantage of recent advances in patterned thin film spectral filters [183], and lensless imaging, to achieve high-resolution snapshot hyperspectral imaging in a small form factor. Our system consists of a tiled spectral filter array placed directly onto the sensor and a randomizing phase mask (i.e. diffuser) placed a small distance away from the sensor, as in the DiffuserCam architecture [8]. The diffuser spatially multiplexes the incoming light, such that each spatial point in the world maps to many pixels on the camera. The spectral filter array then spectrally encodes the incoming light via a structured erasure function. The multiplexing effect of the diffuser allows recovery of scene information from a subset of sensor pixels, so we are able to recover the full spatio-

spectral cube without the loss in resolution that would result from using a non-multiplexing optic, such as a lens.

Our encoding scheme enables hyperspectral recovery in a compact and inexpensive form factor. The spectral filter array can be manufactured directly on the sensor, costing under $5 for both the diffuser and the filter array at scale. A key advantage of our system over previous compact snapshot hyperspectral imagers is that it decouples the spectral and spatial responses, enabling a flexible design in which either contiguous or non-contiguous spectral filters with user-selected bandwidths can be chosen. Given some conditions on scene sparsity and the diffuser randomness, the spectral sampling is determined by the spectral filters and the spatial resolution is determined by the autocorrelation of the diffuser response. This should find use in task-specific/classification applications [181, 43, 122, 89], where one may wish to tailor the spectral sampling to the application by measuring multiple non-contiguous spectral bands, or have higher-resolution spectral sampling for certain bands.

We present theory for our system, simulations to motivate the need for a diffuser, and experimental results from a prototype system, showing:

1. A novel framework for snapshot hyperspectral imaging that combines compressive sensing with spectral filter arrays, enabling compact and inexpensive hyperspectral imaging.
2. Theory and simulations analyzing the system's spatio-spectral resolution for objects with varying complexity.
3. A prototype device demonstrating snapshot hyperspectral recovery on real data from natural scenes.

## Related Work

### Snapshot Hyperspectral Imaging

There have been a variety of snapshot hyperspectral imaging techniques proposed and evaluated over the past decades. Most approaches can be categorized into the following groups: spectral filter array methods, coded aperture methods, speckle-based methods, and dispersion-based methods.

**Spectral filter array methods** use tiled spectral filter arrays on the sensor to recover the spectral channels of interest [116]. These methods can be viewed as an extension of Bayer filters for RGB imaging, since each 'super-pixel' in the tiled array has a grid of spectral filters. As the number of filters increases, the spectral resolution increases and the spatial resolution decreases. For instance, with an 8×8 filter array (64 spectral channels), the spatial resolution is 8× worse in each direction than that of the camera sensor. Demosaicing methods have been proposed to improve upon this in post-processing; however, they rely on intelligently guessing information that is not recorded by the sensor [150]. Recently, photonic crystal slabs have been demonstrated for compact spectroscopy based on random spectral responses (as opposed to traditional passband responses) and extended to hyperspectral imaging through the tiling of the photonic crystal slab pixels [215, 216]. While these methods have high spectral accuracy, they have only been demonstrated in a 10×10 spatial pixel configuration.

Our system uses a spectral filter array, but combines it with a randomizing diffuser in a lensless imaging architecture, allowing us to recover close to the full spatial resolution of the sensor, which is not possible with traditional lens-based methods. Our method uses traditional pass-band spectral filters, but could be extended to photonic crystal slabs and other spectral filter designs.

**Coded aperture methods** use a coded aperture, in combination with a dispersive optical element (e.g. a prism or diffractive grating), in order to modulate the light and encode spatial-spectral information [72, 130, 209, 38]. These systems are able to capture hyperspectral images and videos but tend to be large table-top systems consisting of multiple lenses and optical components. In contrast, our system has a much smaller form factor, requiring only a camera sensor with an attached spectral filter array and a thin diffuser placed close to the sensor.

**Speckle-based methods** use the wavelength dependence of speckle from a random media to achieve hyperspectral imaging. This has been demonstrated for compact spectrometers [172, 41] and extended to hyperspectral imaging [180, 69]. These systems can be compact, since they require only a sensor and scattering media as their optic; however their spectral resolution is limited by the speckle correlation through wavelengths. This is challenging to design for a given application, since the spatial and spectral resolutions are highly coupled. In contrast, our system uses spectral filters that can easily be adjusted for a given application and can be selected to have variable bandwidth or non-uniform spectral sampling.

**Dispersive methods** utilize the dispersion from a prism or diffractive optic to encode spectral information on the sensor. This can be accomplished opportunistically by a prism added to a standard DSLR camera [18]. The resulting system has high spatial resolution, equal to that of the camera sensor, but spectral information is encoded only at the edges of objects in the scene, resulting in a highly ill-conditioned problem and lower spectral accuracy. Other methods use a diffuser (as opposed to a prism) as the dispersive element [75]. This can be more compact than prism-based systems and can have improved spatial resolution when combined with an additional RGB camera [86]. To further improve compactness, [95] uses a single diffractive optic as both the lens and the dispersive element, uniquely encoding spectral information in a spectrally-rotating point spread function (PSF).

Our system uses a lensless architecture and a spectral filter array, together with sparsity assumptions, to reconstruct 3D hyperspectral information across 64 wavelengths. The design is most similar to [95] and achieves a similar compact size; however, our system achieves better spectral accuracy, and the use of the color filter array and diffuser results in more design flexibility, as our spectral and spatial resolutions are decoupled, enabling custom sensors tailored to specific spectral filter bands that do not need to be contiguous.

### Lensless Imaging

Lensless, mask-based imaging systems do not have a main lens, but instead use an amplitude or phase mask in place of imaging optics. These systems have been demonstrated for very compact, small form factor 2D imaging [15, 113, 199, 198]. They are generally amenable

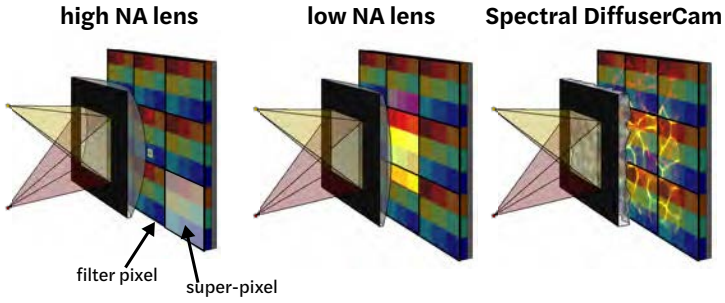


Figure 3.5: **Motivation for multiplexing**: A high-NA lens captures high-resolution spatial information, but misses the yellow point source, since it comes into focus on a spectral filter pixel designed for blue light. A low-NA lens blurs the image of each point source to be the size of the spectral filter's super-pixel, capturing accurate spectra at the cost of poor spatial resolution. Our DiffuserCam approach multiplexes the light from each point source across many super-pixels, enabling the computational recovery of both point sources and their spectra without sacrificing spatial resolution. Note that a simplified 3×3 filter array is shown here for clarity.

to compressive imaging, due to the multiplexing nature of lensless architectures; each point in the scene maps to many pixels on the sensor, allowing a sparse scene to be completely recovered from a subset of sensor pixels [64]. Or, one can reconstruct higher-dimensional functions like 3D [8] or video [9] from a single 2D measurement. In this work, we use diffuser-based lensless imaging to spatially-multiplex light onto a repeated spectral filter array, then reconstruct 3D hyperspectral information. Because of the compressed sensing framework, our spatial resolution is better than the array super-pixel size, despite the missing information due to the array.

## System Design Overview

Our system leverages recent advances in both spectral filter array technology and compressive lensless imaging to decouple the spectral and spatial design. Furthermore, the spectral filter arrays can be deposited directly on the camera sensor. With a diffuser as our multiplexing optic, the system is compact and inexpensive at scale.

To motivate our need for a multiplexing optic instead of an imaging lens, let us consider three candidate architectures: one with a high numerical aperture (NA) lens whose diffraction-limited spot size is matched to the filter pixel size, one with a low-NA lens whose diffraction-limited spot size is matched to the super-pixel size, and finally our design with a diffuser as a multiplexing optic. Figure 3.5 illustrates these three scenarios with a simplified example of a spectral filter array consisting of $3 \times 3$ spectral filters (9 total) repeated horizontally and vertically. Assume that the monochrome camera sensor has square pixels of lateral size $N_{\text{pixel}}$, the spectral filter array has square filters of size $N_{\text{filter}}$, and each $3 \times 3$ block of spectral filters creates a *super-pixel* of size $N_{\text{super-pixel}}$, where $N_{\text{pixel}} < N_{\text{filter}} < N_{\text{super-pixel}}$.

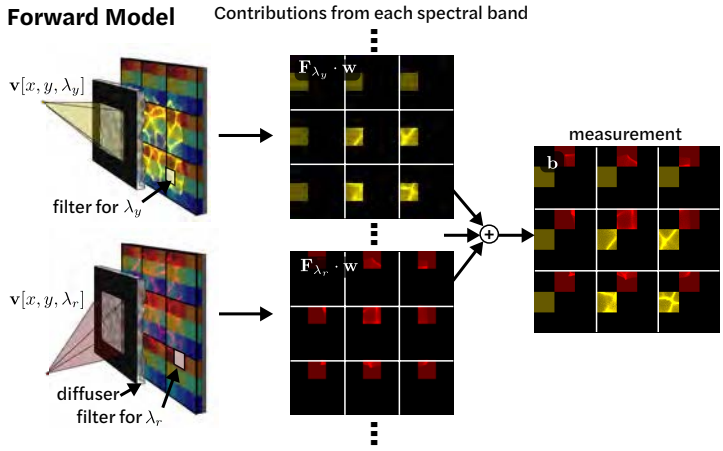


Figure 3.6: **Forward model**. Image formation model for a scene with two point sources of different colors, each with narrow-band irradiance centered at $\lambda_y$ (yellow) and $\lambda_r$ (red). The final measurement is the sum of the contributions from each individual spectral filter band in the array. Due to the spatial multiplexing of the lensless architecture, all scene points $v(x, y, \lambda)$ project information to multiple spectral filters, which is why we can recover a high-resolution hyperspectral cube from a single image, after solving an inverse problem.

In the high-NA lens case, a point source in the scene will be imaged onto a single filter pixel of the sensor, and thus will only be measured if it is within the passband of that filter; otherwise it will not be recorded, Fig. 3.5 (left). In the low-NA lens case, each point source will be imaged to an area the size of the filter array super-pixel, and thus recorded by the sensor correctly, but at the price of low spatial-resolution (matched to the the super-pixel size), Fig. 3.5 (middle). In contrast, a multiplexing optic can avoid the gaps in the measurement of the high-NA lens and achieve better resolution than the low-NA case.

A diffuser multiplexes the light from each point source such that it hits many filter pixels, covering all of the spectral bands. And the spatial resolution of the final image can be on the order of the camera pixel size, provided that conditions for compressed sensing are met, Fig. 3.5 (right). In practice, the spatial resolution of our system will be bounded by the autocorrelation of the point spread function (PSF), as detailed in Sec. 3.2, and the diffuser PSF must span multiple super-pixels to ensure that each point in the world is captured. Since compressive recovery is used to recover a 3D hyperspectral cube from a 2D measurement, the resolution is a function of the scene complexity, as described in Sec. 3.2.

## Imaging Forward Model

Given our design with a diffuser placed in front of a sensor that has a spectral filter array on top of it, in this section we outline a forward model for the optical system, illustrated in Fig. 3.6. This model is a critical piece of our iterative inverse algorithm for hyperspectral reconstruction and will also be used to analyze spatial and spectral resolution.

### Spectral filter model

The spectral filter array is placed on top of an imaging sensor, such that the exposure on each pixel is the sum of point-wise multiplications with the discrete filter function,

$$\mathbf{L}[x, y] = \sum_{\lambda=0}^{K-1} \mathbf{F}_\lambda[x, y] \cdot \mathbf{v}[x, y, \lambda], \tag{3.9}$$

where $\cdot$ denotes point-wise multiplication, $\mathbf{v}[x, y, \lambda]$ is the spectral irradiance incident on the filter array and $\mathbf{F}_\lambda[x, y]$ is a 3D function describing the transmittance of light through the spectral filter for $K$ wavelength bands, which we call the *filter function*. In this model, we absorb the sensor's spectral response into the definition of $\mathbf{F}_\lambda[x, y]$. Our device's filter function is determined experimentally (see Sec 3.2.C) and shown in Fig. 3.7(b). This can be generalized to any arbitrary spectral filter design and does not assume alignment between the filter pixels and the sensor pixels. Here, we focus on the case of a repeating grid of spectral filters, where each 'super-pixel' consists of a set of narrow-band filters. Our device has a 8×8 grid of filters in each super-pixel; Fig. 3.6 illustrates a simplified 3×3 grid, for clarity.

### Diffuser model

The diffuser (a smooth pseudorandom phase optic) in our system achieves spatial multiplexing; this results in a compact form factor and enables reconstruction with spatial resolution better than the super-pixel size via compressed sensing. The diffuser is placed a small distance away from the sensor and an aperture is placed on the diffuser to limit higher angles. The sensor plane intensity resulting from the diffuser can be modeled as a convolution of the scene, $\mathbf{v}[x, y, \lambda]$ with the on-axis PSF, $\mathbf{h}[x, y]$ [113]:

$$\mathbf{w}[x, y, \lambda] = \mathrm{crop}\Big(\mathbf{v}[x, y, \lambda] \overset{[x,y]}{*} \mathbf{h}[x, y])\Big) \tag{3.10}$$

where $\overset{[x,y]}{*}$ represents a discrete 2D linear convolution over spatial dimensions. The crop function accounts for the finite sensor size. We assume that the PSF does not vary with wavelength and validate this experimentally in Sec. 3.2.B. However, this model can be easily extended to include a spectrally-varying PSF, $\mathbf{h}[x, y, \lambda]$ if there is more dispersion across wavelengths.

We assume that objects are placed beyond the hyperfocal distance of the imager, therefore the PSF has negligible depth-variance and a 2D convolutional model is valid [113]. If objects are placed within the hyperfocal distance, a 3D model will be needed to account for the depth-variance of the PSF.

### Combined model

Combining the spectral filter model with the diffuser model, we have the following discrete forward model:

$$\mathbf{b} = \sum_{\lambda=0}^{K-1} \mathbf{F}_\lambda[x,y] \cdot \mathrm{crop}\Big(\mathbf{h}[x,y] \overset{[x,y]}{*} \mathbf{v}[x,y,\lambda]\Big) \tag{3.11}$$

$$= \sum_{\lambda=0}^{K-1} \mathbf{F}_\lambda[x,y] \cdot \mathbf{w}[x,y,\lambda] \tag{3.12}$$

$$= \mathbf{A}\mathbf{v}. \tag{3.13}$$

The linear forward model is represented by the combined operations in matrix $\mathbf{A}$. Figure 3.6 illustrates the forward model for several point sources, showing the intermediate variable $\mathbf{w}[x,y,\lambda]$, which is the scene convolved with the PSF, before point-wise multiplication by the filter function. The final image is the sum over all wavelengths.

## Hyperspectral Reconstruction

To recover the hyperspectral datacube from the 2D measurement, we must solve an underdetermined inverse problem. Since our system falls within the framework of compressive sensing due to our incoherent, multiplexed measurement, we use $l_1$ minimization. We use a weighted 3D total variation (3DTV) prior on the scene, as well as a non-negativity constraint, and a low-rank prior on the spectrum. This can be written as:

$$\hat{\mathbf{v}} = \underset{\mathbf{v}\geq 0}{\arg\min} \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{v}\|_2^2 + \tau_1\|\nabla_{xy\lambda}\mathbf{v}\|_1 + \tau_2\|\mathbf{v}\|_*, \tag{3.14}$$

where $\nabla_{xy\lambda} = [\nabla_x \nabla_y \nabla_\lambda]^T$ is the matrix of forward finite differences in the $x$, $y$, and $\lambda$ directions, $\|\cdot\|_*$ represents the nuclear norm, which is the sum of singular values. $\tau_1$ and $\tau_2$ are the tuning parameters for the 3DTV prior and low-rank priors, respectively. We use the fast iterative shrinkage-thresholding algorithm (FISTA) [20] with weighted anisotropic 3DTV to solve this problem according to [100].

## Implementation Details

We built a prototype system using a CMOS sensor, a hyperspectral filter array provided by Viavi Solutions (Santa Rosa, CA)[183], and an off-the-shelf diffuser (Luminit 0.5°) placed 1cm away from the sensor. The sensor has 659×494 pixels (with a pixel pitch of 9.9$\mu m$), which we crop down to 448×320 to match the spectral filter array size. The spectral filter array consists of a grid of 28×20 super-pixels, each with an 8×8 grid of filter pixels (64 total, spanning the range 386–898nm). Each filter pixel is 20$\mu m$ in size, covering slightly more than 4 sensor pixels. The alignment between the sensor pixels and the filter pixels is unknown, requiring a calibration procedure (detailed in Sec. 3.23.2). The exposure time is adjusted for each image, ranging from 1ms-13ms, which is short enough for video-rate acquisition. The computational reconstruction typically takes 12–24 minutes (for 500–1000 iterations) on an RTX 2080-Ti GPU using MATLAB.

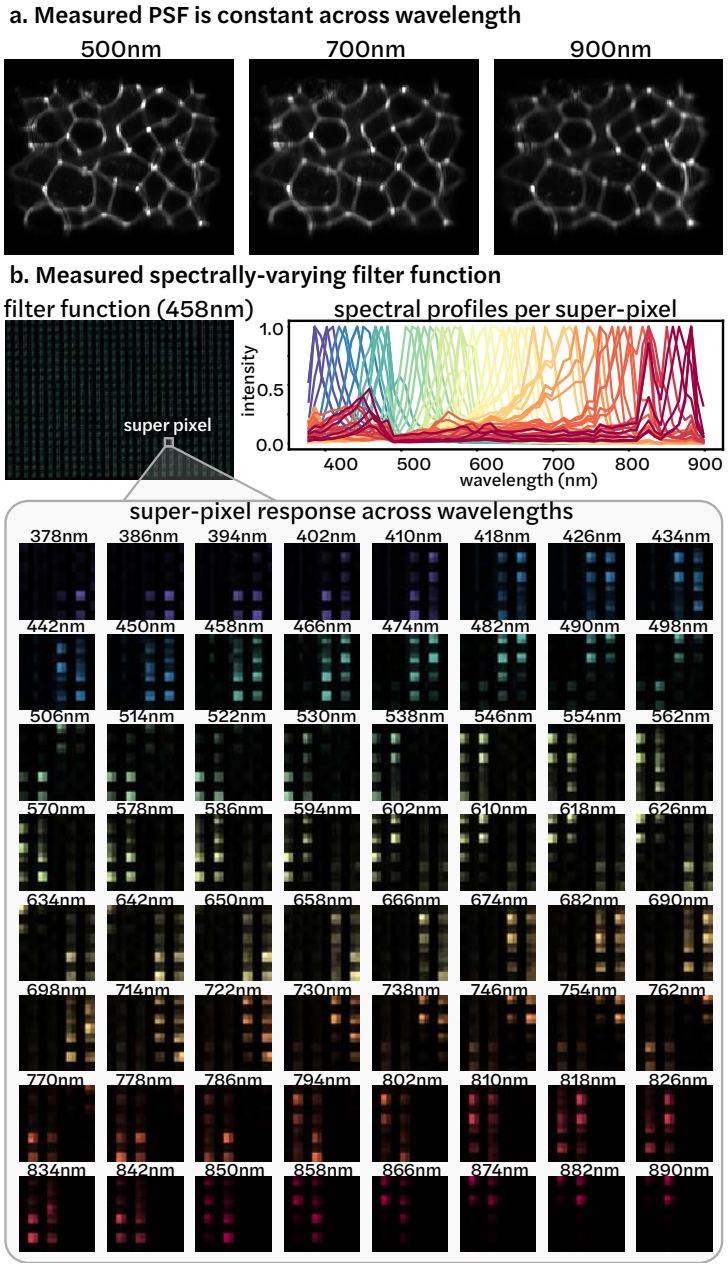


Figure 3.7: **Experimental calibration of Spectral DiffuserCam**. (a) The caustic PSF (contrast-stretched and cropped), before passing through the spectral filter array, is similar at all wavelengths. (b) The spectral response with the filter array only (no diffuser). (Top left) Full measurement with illumination by a 458nm plane wave. The filter array consists of 8×8 grids of spectral filters repeating in 28×20 super-pixels. (Top right) Spectral responses of each of the 64 color channels. (Bottom) Spectral response of a single super-pixel as illumination wavelength is varied with a monochromater.

### Filter Function Calibration

To calibrate the filter function ($\mathbf{F}_\lambda[x, y]$ in Eqn. 3.11), including the spectral sensitivity of both the sensor and the spectral filter array, we use a Cornerstone 130 1/3m motorized monochromator (Model 74004). The monochromater creates a narrow-band source of 5nm full-width at half-maximum (FWHM) and we measure the filter response (without the diffuser) while sweeping the source by 8nm increments from 386nm to 898nm. The result is shown in Fig. 3.7(b).

### PSF Calibration

We also need to calibrate the diffuser response by measuring the diffuser PSF pattern without the spectral filter array. Because the diffuser is relatively smooth with large features (relative to the wavelength of light), the PSF remains relatively constant as a function of wavelength, as shown in Fig. 3.7(a). Hence, we only need to calibrate for a single wavelength by capturing a single point source calibration image [8]. However, this is not trivial because the spectral filter array is bonded to the sensor and cannot be removed easily. In our setup, we instead take advantage of the fact that our filter array is smaller than our sensor, so we can measure the PSF using the edges of the raw sensor, by shifting the point source to scan the different parts of the PSF over the raw sensor area and stitching the sub-images together. In a system where the filter size is matched to the sensor, this trick will not be possible, but an optimization-based approach could be developed to recover the PSF from measurements.

### System Non-idealities

Our reconstruction quality and spectral resolution are limited by two non-idealities in our system. First, our camera development board performs an undefined and uncontrollable non-linear contrast-stretching to all images. This makes the measurement non-linear and impedes our imaging of dim objects (since the camera performs a larger contrast stretching for dimmer images). Further, our spectral calibration may have errors, since each calibration image cannot be normalized by the intensity of light hitting the sensor. This may cause certain wavelength bands to appear brighter or dimmer than they should in our spectral reconstructions. A better camera board without automatic contrast stretching should fix this problem and provide more quantitative spectral profile reconstructions in the future.

Second, we used a simplified spectral calibration in which we measured the response with uniform spectral sampling, instead of at the true wavelengths of the filters. Due to the mismatch between our calibration scheme (measured every 8nm with constant bandwidth) and the actual spectral filters (center wavelengths spaced 5–12nm apart with bandwidths between 6–23nm), sometimes our calibration wavelengths fall between two filters, resulting in an ambiguity. Given this non-ideal calibration, our effective spectral bands are limited to 49 bands, instead of 64. In our results, we show all 64 bands, but note that some will have overlapping spectral responses. In the future, we will calibrate at the design wavelengths of the filter to fix this issue. Further, the deposition of the spectral filters directly on-top of the

camera pixels (requiring precise placement during the manufacturing stage) would alleviate the need for this calibration entirely.

## Resolution Analysis

Here, we derive our theoretical resolution and experimentally validate it with our prototype system. First, we discuss spectral resolution, which is set by the filter bandwidths, and then we compute the expected two-point spatial resolution, based on the PSF autocorrelation. Since our resolution is scene-dependent, we expect the resolution to degrade with scene complexity. To characterize this, we present theory for multi-point resolution based on the condition number analysis introduced in [8]. We compare our system against those with a high-NA and low-NA lens instead of a diffuser. Our results demonstrate two-point spatial resolution of ~0.19 super-pixels and multi-point spatial resolution of ~0.3 super-pixels for 64 spectral channels ranging from 386–898nm.

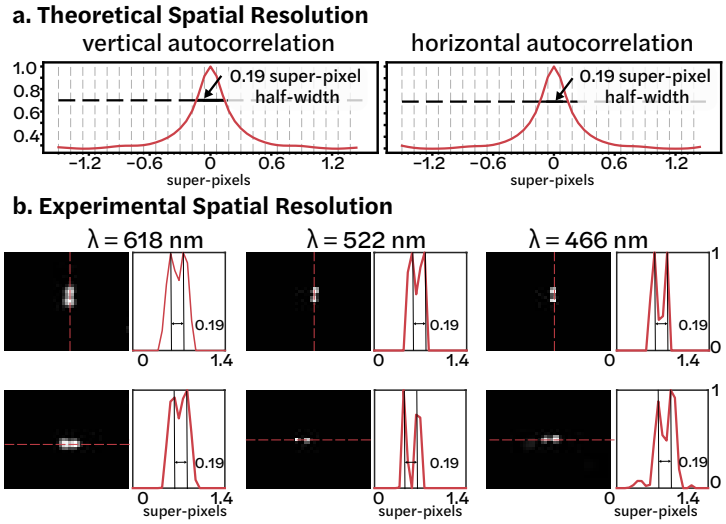


Figure 3.8: **Spatial resolution analysis**. (a) The theoretical resolution of our system, defined as the half-width of the autocorrelation peak at 70% its maximum value, is 0.19 super-pixels. (b) Experimental two-point reconstructions demonstrate 0.19 super-pixel resolution across all wavelengths (slices of the reconstruction shown here), matching the theoretical resolution.

### Spectral Resolution

Spectral resolution is determined by the spectral channels of the filter array. As such, we expect to be able to resolve the 64 spectral channels present in our spectral filter array. The filters have an average spacing of 8nm across a 386–898nm range with bandwidths between 6–23nm. To validate our spectral resolution, we scan a point source across those wavelengths

using a monochrometer. Figure 3.9 shows a sampling of spectral reconstructions overlaid on top of each other, with the shaded blocks indicating the ground-truth monochrometer spectra. Our reconstructions all match the ground-truth peaks within 5nm of the true wavelength. The small red peaks around 400nm are artifacts from the monochrometer, which emitted a 2nd peak around 400nm for the longer wavelengths.

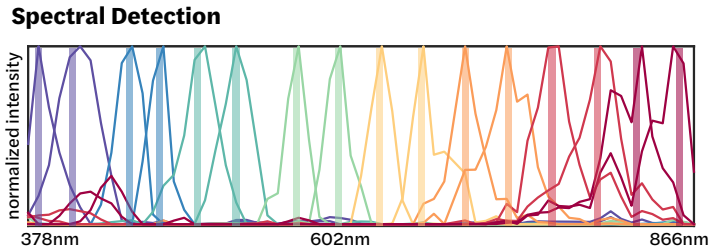


Figure 3.9: **Spectral resolution analysis**. Sample spectra from hyperspectral reconstructions of narrow-band point sources, overlaid on top of each other, with shaded lines indicating the ground-truth. For each case, the recovered spectral peak matches the true wavelength within 5nm.

### Two-point Spatial Resolution

Spatial resolution of our system, in terms of the two-point resolution, will be bounded by that of a lensless imager with the diffuser only (without the spectral filter array). The expected resolution can be defined as the autocorrelation peak half-width at 70% the maximum value [113], Fig. 3.8(a). For our system, this is ~3 sensor pixels, or 0.19 super-pixels. To experimentally measure the spatial resolution of our system, we image two point sources at three different wavelengths (618 nm, 522 nm, 466 nm). The reconstructions in Fig. 3.8 show that we can resolve two point sources that are 0.19 super-pixels apart for each wavelength and orientation, as determined by applying the Rayleigh criterion. This demonstrates that our system achieves sub-super-pixel spatial resolution, consistent with the expected resolution that would be achieved without the spectral filter array.

### Multi-point resolution

Because our image reconstruction algorithm contains nonlinear regularization terms, our reconstruction resolution will be object dependent. Hence, two-point resolution measurements are not sufficient for fully characterizing the system resolution, and should be considered a *best case* scenario. To better predict real-world performance, we perform a local condition number analysis, as introduced in [8], that estimates resolution as a function of object complexity. The local condition number is a proxy for how well the forward model can be inverted, given known support, and is useful for systems such as ours in which the full $\mathbf{A}$ matrix is never explicitly calculated [36].

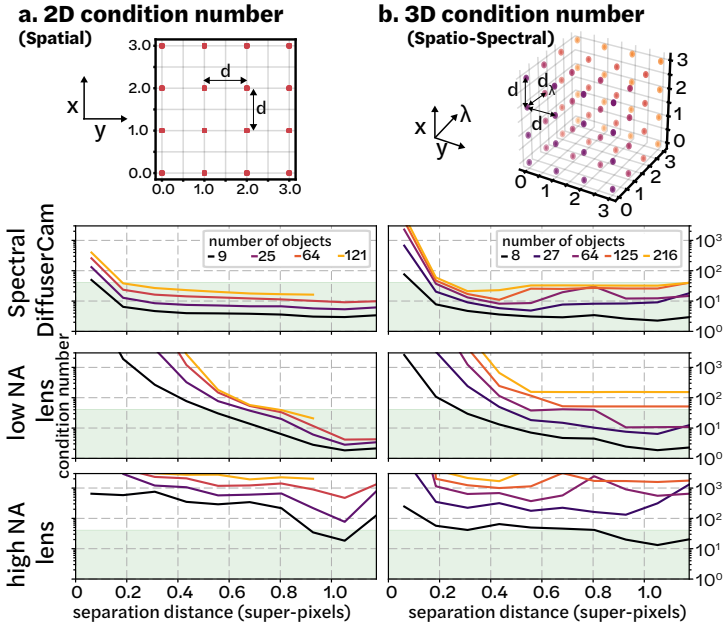


Figure 3.10: **Condition number analysis** for Spectral DiffuserCam, as compared to a low-NA or high-NA lens. (a) Condition numbers for the 2D spatial case (single spectral channel) are calculated by generating different numbers of points on a 2D grid, each with separation distance $d$. (b) Condition numbers for the full spatio-spectral case are calculated on a 3D grid. A condition number below 40 is considered to be good (shown in green). The diffuser has a consistently better performance for small separation distances than either the low-NA or the high-NA lens. The diffuser can resolve objects as low as 0.3 super-pixels apart for more complex scenes, whereas the low-NA lens requires larger separation distances and the high-NA lens suffers errors due to gaps in the measurement.

The local condition number theory states that given knowledge of the *a priori* support of the scene, $\mathbf{v}$, we can form a sub-matrix consisting only of columns of $\mathbf{A}$ corresponding to the non-zero voxels. The reconstruction problem will be ill-posed if any of the sub-matrices of $\mathbf{A}$ are ill-conditioned, which can be quantified by the condition number of the sub-matrices. The worst-case condition number will be when sources are near each other, therefore we compute the condition number for a group of point sources with a separation varying by an integer number of voxels and repeat this for increasing numbers of point sources.

In Fig. 3.10, we calculate the local condition number for two cases: the 2D spatial reconstruction case, considering only a single spectral channel, and the 3D case, considering points with varying spatial and spectral positions. For comparison, we also simulate the condition number for a low-NA and high-NA lens, as introduced in Sec. 3.2. The results show that our diffuser design has a consistently lower condition number than either the low- or high-NA lens, having a condition number below 40 for separation distances of greater than ~0.3 super-pixels. The low-NA lens needs a separation distance closer to ~1 super-

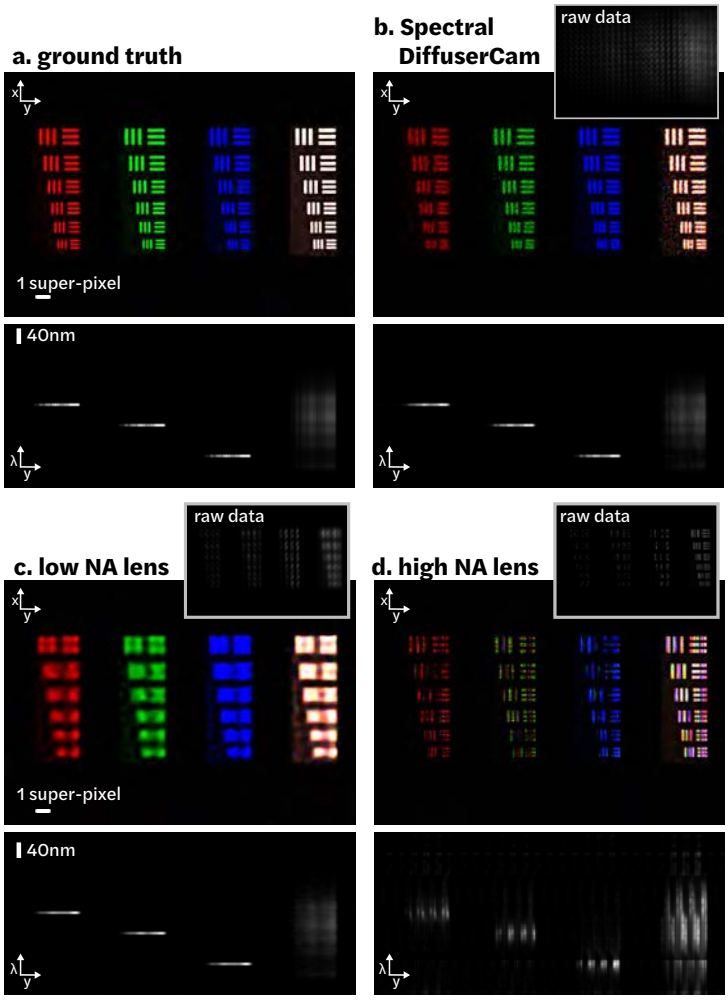


Figure 3.11: **Simulated hyperspectral reconstructions** comparing our Spectral DiffuserCam result with alternative design options. (a) Resolution target with different sections illuminated by narrow-band 634nm (red), 570nm (green), 474nm (blue), and broadband (white) sources. (b) Reconstruction of the target by Spectral DiffuserCam, (c) a low-NA lens design, and (d) a high-NA lens design, each showing the raw data, false-colored reconstruction and $\lambda y$ sum projection. The diffuser achieves higher spatial resolution and better accuracy than the low-NA and the high-NA lens.

pixel, as expected, and the high-NA lens has an erratic condition number due to the missing information in the measurement.

From this analysis, we can see that, beyond 0.3 super-pixels separation, the condition number for the diffuser does not get arbitrarily worse for increasing scene complexity. Thus, our expected spatial resolution is approximately 0.3 super-pixels.

### Simulated Resolution Target Reconstruction

Next, we validate the results of our condition number analysis through simulated reconstructions of a resolution target with different spatial locations illuminated by different sources (red, green, blue and white light), as shown in Fig. 3.11. For each simulation, we add Gaussian noise with a variance of $1 \times 10^{-5}$ and run the reconstruction for 2,000 iterations of FISTA with 3DTV. Our system resolves features that are 0.3 super-pixels apart, whereas the low-NA lens can only resolve features that are roughly 1 super-pixel apart and the high-NA lens results in gaps, validating our predicted performance.

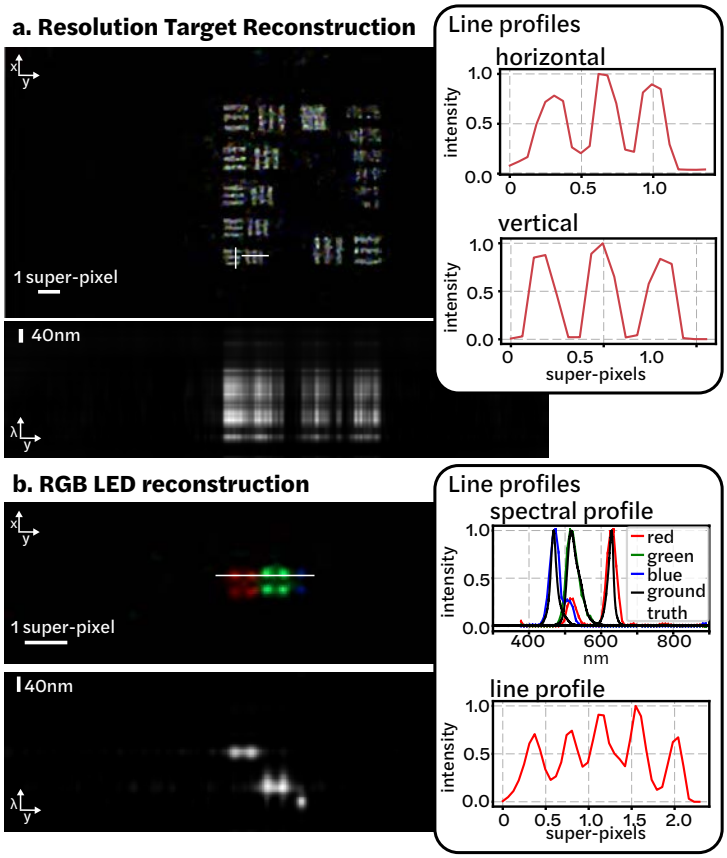


Figure 3.12: **Experimental Results**. (a) Experimental reconstruction of a broadband resolution target, showing the $xy$ sum projection (top) and $\lambda y$ sum projection (bottom), demonstrating spatial resolution of 0.3 super-pixels. (b) Experimental reconstruction of 10 multi-colored LEDs in a grid with ~0.4 super-pixels spacing (four red LEDs on left, four green in middle, two blue at right). We show the $xy$ sum projection (top) and $\lambda y$ sum projection (bottom). The LEDs are clearly resolved spatially and spectrally, and spectral line profiles for each color LED closely match the ground truth spectra from a spectrometer.

## Experimental Results

We start with experimental reconstructions of simple objects with known properties—a broadband USAF resolution target displayed on a computer monitor, and a grid of RGB LEDs (Fig. 3.12). We resolve points that are ∼.3 super-pixels apart, which matches our expected multi-point resolution based on the condition number analysis above. For the RGB LED scene, the ground truth spectral profiles of the LEDs are measured using a spectrometer, and our recovered spectral profile closely matches the ground truth, as shown in Fig. 3.12(b).

Next, we show reconstructions of more complex objects, either displayed on a computer monitor or illuminated with two halogen lamps (Figure 3.13). We plot the ground truth spectral line profiles, as measured by a Thorlabs CCS200 spectrometer, from four points in the scene, showing that we can accurately recover the spectra. A reference RGB scene is shown for each image, demonstrating that the reconstructions spatially match the expected scene.

## Discussion

A key advantage of our design over previous work is its flexibility to choose the spectral filters in order to tailor the system to a specific application. For example, one can non-linearly sample a wide range of wavelengths (which is difficult with many previous snapshot hyperspectral imagers). In future, we plan to design implementations specific to various task-based applications, which could make hyperspectral imaging more easily adopted, especially since the price is several orders-of-magnitude lower than currently available hyperspectral cameras.

Currently, we experimentally achieve a spatial resolution of ∼0.3 super-pixels, or 5 sensor pixels. In future designs, we should be able to achieve the full sensor resolution (along with better quality reconstructions) by optimizing the randomizing optic, instead of using an off-the shelf diffuser. This could be achieved by end-to-end optical design [188, 168].

Our system has two main limitations: light-throughput and scene-dependence. Due to the use of narrow-band spectral filters, much of the light is filtered out by the filters. This provides good spectral accuracy and discrimination, but at the cost of low light throughput. In addition, since the light is spread by the diffuser over many pixels, the signal-to-noise ratio (SNR) is further decreased. Hence, our imager is not currently suitable for low-light conditions. This light-throughput limitation can be mitigated in the future by the use of photonic crystal slabs instead of narrowband filters, in order to increase light-throughput while maintaining spatio-spectral resolution and accuracy [216]. In addition, end-to-end design of both the spectral filters and the phase mask should improve efficiency, since application-specific designs can use only the set of wavelengths necessary for a particular task, without sampling the in-between wavelengths. Reducing the number of spectral bands improves both light throughput (because more sensor area will be dedicated to each spectral band) and spatial resolution (because the super-pixels will be smaller).

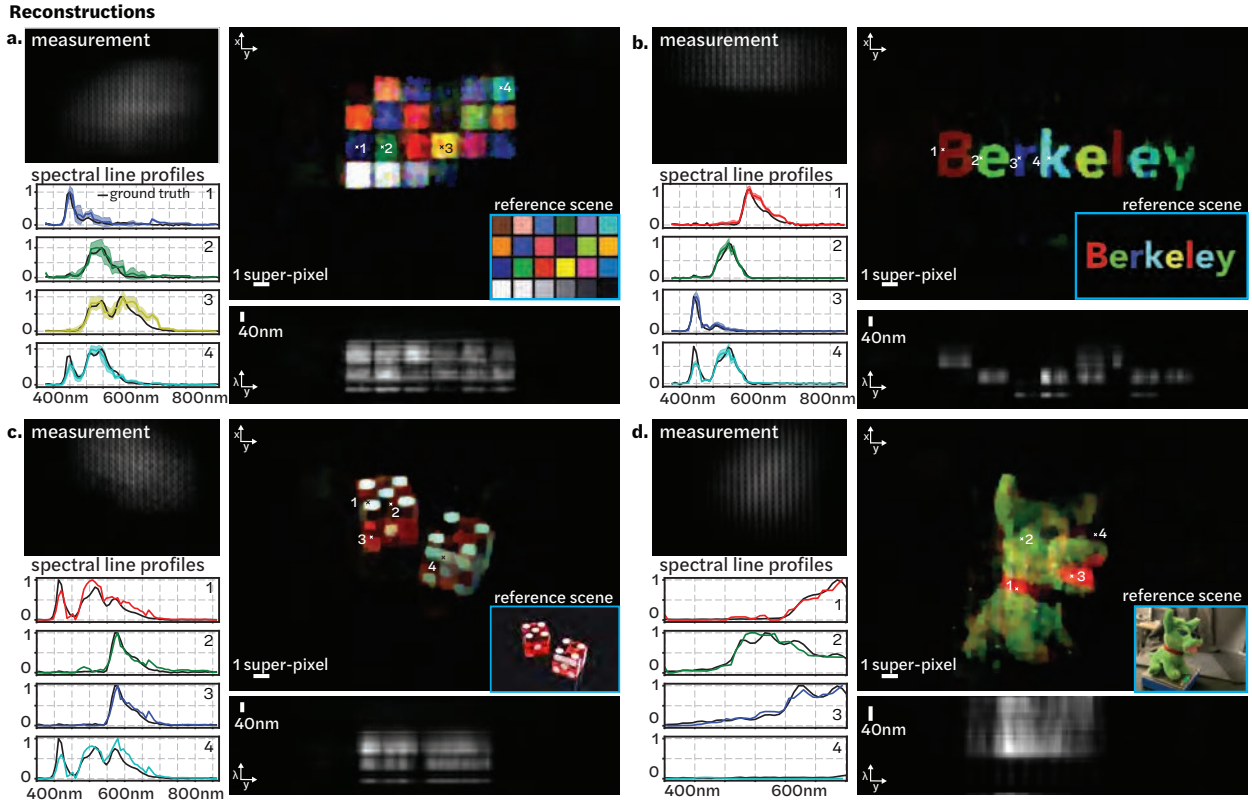


Figure 3.13: **Experimental hyperspectral reconstructions**. (a-c) Reconstructions of color images displayed on a computer monitor and (d) Thorlabs plush toy placed in front of the imager and illuminated by two Halogen lamps. The raw measurement, false color images, $x\lambda$ sum projections and spectral line profiles for four spatial points are shown for each scene. The ground truth spectral line profiles, measured using a spectrometer, are plotted in black for reference. Spectral line profiles in (a,b) show the average and standard deviation spectral profiles across the area of the box or letter in the object, whereas (c-d) show a line profile from a single spatial point in the scene.

Our second limitation is scene-dependence, as our reconstruction algorithm relies on object sparsity (e.g. sparse gradients). Because of the non-linear regularization term, it is difficult to predict performance, and one might suffer artifacts if the scene is not sufficiently sparse. Recent advances in machine learning and inverse problems seek to provide better signal representations, enabling the reconstruction of more complicated, denser scenes [134, 24]. In addition, machine learning could be useful in speeding up the reconstruction algorithm [156] as well as potentially utilizing the imager more directly for a higher-level task, such as classification [55].

## Conclusion

Our work presents a new hyperspectral imaging modality that combines a color filter array and lensless imaging techniques for an ultra-compact and inexpensive hyperspectral camera. The spectral filter array encodes spectral information onto the sensor and the diffuser multiplexes the incoming light such that each point in the world maps to many spectral filters. The multiplexed nature of the measurement allows us to use compressive sensing to reconstruct high spatio-spectral resolution from a single 2D measurement. We provided an analysis for the expected resolution of our imager and experimentally characterized the two-point and multi-point resolution of the system. Finally, we built a prototype and demonstrated reconstructions of complex spatio-spectral scenes, achieving up to 0.19 super-pixel spatial resolution across 64 spectral bands.

# Chapter 4

# Physics-informed Supervised Learning for 2D Lensless Photography

In this chapter[1], we demonstrate how physics-informed machine learning can be used to improve reconstructions for 2D lensless computational cameras. We showcase how classic and deep methods can be combined by taking a classic method, and unrolling it. Through this process, we can create a bounded, differentiable reconstruction method that can be optimized to produce the best image quality within a bounded number of iterations. Furthermore, our differentiable unrolled network can be easily combined with existing deep networks to further improve performance. We use a dataset of real experimentally captured lensed and lensless images to train our dataset, showing a 20× speedup and 3× improvement in perceptual image quality over classic methods.

## 4.1 Related Work

Image reconstruction methods for lensless cameras fall into two general categories: single-step and iterative reconstructions. Single-step reconstructions can be fast, but often require custom fabricated masks that must be carefully aligned to the sensor [14, 99, 73, 199]. In addition, it is difficult to incorporate priors and leverage compressed sensing in single-step reconstructions. Iterative reconstructions are much slower, but do not impose stringent restrictions on the mask itself, generally produce better results, and allow priors to be used [113, 191]. However, due to imperfect system modeling, these methods may still give significant reconstruction artifacts. Additionally, the high complexity of the computation precludes interactive previewing of the scene and requires expensive, bulky compute hardware. In this work, we focus on iterative methods, improving both the image quality and speed with a new reconstruction framework that incorporates the advantages of both deep learning and physical models, making lensless cameras more practical for everyday imaging.

[1]This chapter is based on the published journal paper titled "Learned reconstructions for practical mask-based lensless imaging" and is joint work with Joshua Yurtsever, Grace Kuo, Nick Antipa, Kyrollos Yanny, and Laura Waller [156].

The classical approach to image recovery is to use convex optimization to iteratively minimize a loss function [20, 28] consisting of a data-fidelity term and an optional hand-picked regularization term. The data-fidelity term enforces that the recovered image, with the known imaging model applied to it, matches the measurement. The regularization term enforces prior knowledge of image statistics (e.g. non-negative, sparse gradients) and serves to regularize ill-conditioned problems. Iterative approaches are interpretable, but are sensitive to reconstruction artifacts due to model mismatch, calibration errors, hand-tuned parameters, and hand-picked regularizers which are not necessarily representative of the data. Each of these contributes to reconstruction artifacts and degrades image quality. Furthermore, these methods can take hundreds to thousands of iterations to converge, which is often too slow for real-time imaging.

Recently, deep learning-based methods for image reconstruction have risen in popularity. In deep methods, a convolutional neural network (CNN) is used for image reconstruction [127, 128, 163]. Networks have hundreds of thousands of parameters which are updated using large datasets of image pairs. These networks are able to learn complex scene statistics, but do not incorporate any prior knowledge about the image formation process. Compared to iterative methods, deep learning-based methods are hard to interpret, do not have convergence guarantees, and have no structured way to incorporate knowledge of the imaging system physics.

Unrolled optimization represents a middle-ground between classic and deep methods. In unrolled optimization, a fixed number of iterations from a classic algorithm is interpreted as a deep network, with each iteration serving as a layer in the network. In each layer, if the parameters of the algorithm are differentiable with respect to the output, they can be optimized for a given loss function through backpropagation. In this framework, the sparsifying filters, hyper-parameters, or shrinkage function can be learned from the training examples [80, 184]. Unrolled optimization has shown promising results for image denoising [55, 56], sparse coding [80], and MRI reconstructions [193].

Here, we unroll the iterative *alternating direction method of multipliers* (ADMM) algorithm with a variable splitting specific for lensless imaging [28, 8]. This allows us to incorporate knowledge of the image formation process into the neural network as well as learn the network parameters based on the data. To train our network, we experimentally capture a large dataset of lensed and lensless images (Fig. 4.1). We train our network on a perceptual similarity metric in order to produce images that are visually similar to those from our ground truth lensed camera. We present several variations of networks along the spectrum between classic methods and deep methods, by varying the number of trainable parameters (Fig. 4.2). Specifically, we introduce three architectures, Le-ADMM, Le-ADMM*, and Le-ADMM-U, each with increasing numbers of trainable parameters, explained in detail in Sec. 4.4. All of our networks have a bounded compute that can be adjusted according to the application. The networks trade-off data fidelity and image perceptual quality, producing more visually appealing images at the price of decreased data-fidelity.

We test our network using DiffuserCam [113] as our prototypical lensless camera, built with off-the-shelf components and a low-end camera sensor. Although our network is trained using images from a computer monitor, we demonstrate the generalization of our network to

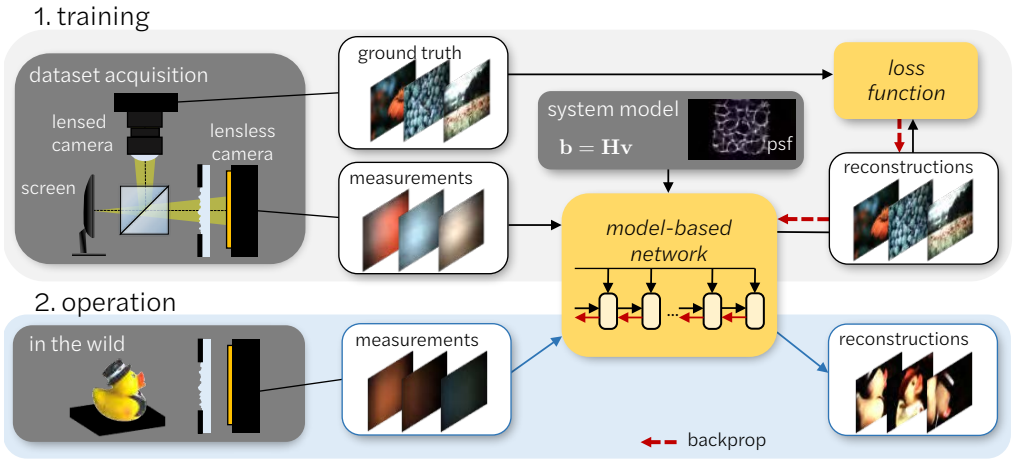


Figure 4.1: **Overview of our imaging pipeline**. During training, images are displayed on a computer screen and captured simultaneously with both a lensed and a lensless camera to form training pairs, with the lensed images serving as ground truth labels. The lensless measurements are fed into a model-based network which incorporates knowledge about the physics of the imager. The output of the network is compared with the labels using a loss function and the network parameters are updated through backpropagation. During operation, the lensless imager takes measurements and the trained model-based network is used to reconstruct the images, providing a large speedup in reconstruction time and an improvement in image quality.

measurements of natural objects taken in the wild. We believe that this exploratory work shows the promise of using unrolled neural networks for lensless imaging, and our results suggest the utility of combining knowledge of the physics together with deep learning for the best performance.

Our contributions include:

1. A bounded-time trainable network architecture that incorporates knowledge of the physical model for lensless imaging.
2. An experimental dataset of 25,000 aligned lensed and lensless image pairs taken using a beamsplitter and computer screen.
3. A demonstration of 20× speedup and 3× improvement in perceptual similarity for lensless imaging reconstructions on an experimental system.
4. Generalization of the network to images taken in the wild on a prototype lensless camera.

## 4.2 Lensless Imaging Forward Model

First, we describe our lensless imaging forward model for DiffuserCam. Based on this, we formulate our traditional model-based reconstruction (Sec. 4.3), before moving on to

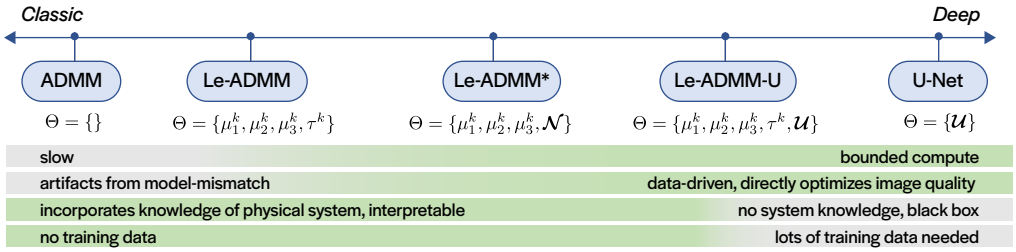


Figure 4.2: **Networks on a scale from classic to deep**. We will present several networks specifically designed for lensless imaging (Le-ADMM, Le-ADMM*, and Le-ADMM-U). We compare these to classic approaches, which have no learnable parameters, and to purely deep methods which do not include any knowledge of the imaging model. We will show the utility of using an algorithm in this middle range compared to a purely classic or deep method. $\Theta$ summarizes the parameters that are learned for each network as discussed in Section 4.4.

modifications that span the spectrum from model-based to deep learning-based algorithms (Fig. 4.2) in Sec. 4.4.

DiffuserCam [8, 113] is a compact, easy-to-build imaging system that consists only of a diffuser (a transparent phase mask with pseudo-random slowly varying thickness) placed a few millimeters in front of a standard image sensor (see Fig. 4.1). Light from a point source in the scene is refracted by the diffuser to create a high-contrast caustic pattern on the sensor, which is the point spread function (PSF) of the system (Fig. 4.1). Since the diffuser is thin, the PSF can be modeled as shift-invariant: a lateral shift of the point source in the scene causes a translation of the PSF in the opposite direction. We model the scene as a collection of point sources with varying color and intensity. Assuming all points are incoherent with each other, the sensor measurement, $\mathbf{b}$, can be described as:

$$\begin{aligned} \mathbf{b}(x,y) &= \text{crop}[\mathbf{h}(x,y) * \mathbf{x}(x,y)] \\ &= \mathbf{CHx}, \end{aligned} \tag{4.1}$$

where $\mathbf{h}$ is the system PSF, $\mathbf{x}$ represents the scene, and $(x, y)$ are the sensor coordinates. Here, $*$ denotes 2D discrete linear convolution, which returns an array that is larger than both the scene and the PSF. Therefore, a crop operation restricts the output to the physical sensor size. This relation is represented compactly in matrix-vector notation with crop denoted as $\mathbf{C}$ and convolution with the PSF denoted as $\mathbf{H}$. Equation (4.1) is computed separately for each color channel.

Our goal is to recover the scene, $\mathbf{x}$, from the measurement $\mathbf{b}$. We assume the PSF is known, as it can easily be measured experimentally with an LED point source [8]. Traditional model-based methods for recovering $\mathbf{x}$ solve a regularized optimization problem of the following form:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\geq 0}{\arg\min} \frac{1}{2}\|\mathbf{b} - \mathbf{C}\mathbf{H}\mathbf{x}\|_2^2 + \tau\|\Psi\mathbf{x}\|_1, \tag{4.2}$$

where $\Psi$ is a sparsifying transform, such as finite differences for total variation (TV) denoising, and $\tau$ is a tuning parameter that adjusts the sparsity level.

## 4.3 Model-based Inverse Algorithm

The traditional model-based inverse solver relies on the known physics of the forward model to solve Eq. (4.2), minimizing the difference between the actual and predicted measurements, while satisfying any additional constraints. This problem can be solved efficiently by ADMM [28] with a variable splitting that leverages the structure of the problem [8]. In ADMM, the problem is reformulated as:

$$\begin{aligned} \hat{\mathbf{x}} = \underset{w\geq 0,u,v}{\arg\min} \frac{1}{2}\|\mathbf{b} - \mathbf{C}v\|_2^2 + \tau\|u\|_1, \\ s.t.\ v = \mathbf{H}\mathbf{x}, u = \Psi\mathbf{x}, w = \mathbf{x}. \end{aligned} \tag{4.3}$$

This variable splitting allows closed-form updates for each step, as derived in [8]. The update equations in each iteration become:

$$\begin{aligned} u^{k+1} &\leftarrow \mathcal{T}_{\tau/\mu_2}(\mathbf{\Psi}\mathbf{x}^k + \alpha_2^k/\mu_2) && \text{sparsifying soft-threshold} \\ v^{k+1} &\leftarrow (\mathbf{C}^\mathbf{T}\mathbf{C} + \mu_1 I)^{-1}(\alpha_1^k + \mu_1\mathbf{H}\mathbf{x}^k + \mathbf{C}^\mathbf{T}\mathbf{b}) && \text{least-squares update} \\ w^{k+1} &\leftarrow \max(\alpha_3^k/\mu_3 + \mathbf{x}^k, 0) && \text{enforce non-negativity} \\ \mathbf{x}^{k+1} &\leftarrow (\mu_1\mathbf{H}^\mathbf{T}\mathbf{H} + \mu_2\mathbf{\Psi}^\mathbf{T}\mathbf{\Psi} + \mu_3 I)^{-1} r^k && \text{least-squares update} \\ \alpha_1^{k+1} &\leftarrow \alpha_1^k + \mu_1(\mathbf{H}\mathbf{x}^{k+1} - v^{k+1}) && \text{dual for } v \\ \alpha_2^{k+1} &\leftarrow \alpha_2^k + \mu_2(\mathbf{\Psi}\mathbf{x}^{k+1} - u^{k+1}) && \text{dual for } u \\ \alpha_3^{k+1} &\leftarrow \alpha_3^k + \mu_3(\mathbf{x}^{k+1} - w^{k+1}) && \text{dual for } w \end{aligned}$$

where $r^k = ((\mu_3 w^{k+1} - \alpha_3^k) + \mathbf{\Psi}^\mathbf{T}(\mu_2 u^{k+1} - \alpha_2^k) + \mathbf{H}^\mathbf{T}(\mu_1 v^{k+1} - \alpha_1^k))$. Here, $\alpha_1$, $\alpha_2$, and $\alpha_3$ are the Lagrange multipliers, or dual variables, respectively associated with $u$, $v$, and $w$, and $\mu_1$, $\mu_2$, and $\mu_3$ are scalar penalty parameters. $\mathcal{T}_{\tau/\mu_2}$ denotes vectorial soft-thresholding with parameter $\tau/\mu_2$.

This traditional method is based on the physical model of the imaging system (Eq. (4.1)) and requires no additional calibration data beyond the PSF. However, it depends heavily on hand-chosen values, such as the sparsifying transform $\Psi$ and its associated parameter, $\tau$. The optimization parameters, $\mu_1$, $\mu_2$, and $\mu_3$, are either hand-tuned or auto-tuned based on the primal and dual residuals at each iteration [28]. The method performs well under

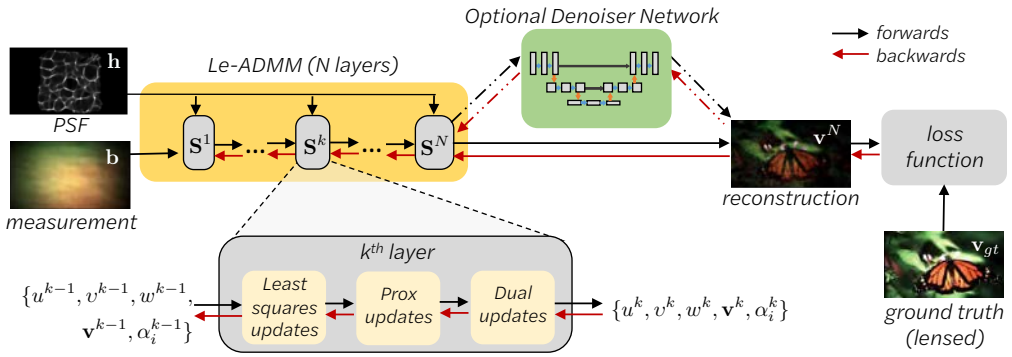


Figure 4.3: **Model-based network architecture**. The input measurement and the calibration PSF are first fed into N layers of unrolled Le-ADMM. At each layer, the updates corresponding to $\mathbf{S}^{k+1}$ in Eq. (4.4) are applied. The output of this can be fed into an optional denoiser network. The network parameters are updated based on a loss function comparing the output image to the lensed image. Red arrows represent backpropagation through the network parameters.

correctly chosen sparsifying transforms and with the proper hand-tuned parameters. However, in practice ADMM takes hundreds of iterations to converge and produces images with reconstruction artifacts. In the next section, we will outline how we unroll ADMM into a neural network in order to learn the hyper-parameters from the data and seamlessly interface with existing deep learning pipelines.

## 4.4 Learned Reconstruction Networks

Next, we present several variations of neural networks that jointly incorporate known physical models and deep learning principles. Each network is based on unrolling the iterative ADMM algorithm, such that each iteration comprises a layer of the network, with the tunable parameters learned from the training data. Thus, the physical model is inherently built into the network architecture, making it more efficient.

We present three variations of networks, each having a different number of learned parameters. Learned ADMM (Le-ADMM) has trainable tuning and hyper-parameters. Le-ADMM* extends Le-ADMM by adding a trainable CNN instead of a hand-tuned sparsifying transform. Finally, Le-ADMM-U adds a trainable deep denoiser based on a CNN as the last layer of the Le-ADMM network, learning both the hyper-parameters of Le-ADMM as well as the denoiser. Figure 4.2 summarizes these methods and where they fall on a scale from classic to deep, and the following sections describe them in detail. Each method has progressively more trainable parameters, and therefore needs a larger training dataset. All networks use 5 iterations of unrolled ADMM, in order to target a 20× speed improvement, which would speed up each reconstruction from 1.5 s to 75 ms, giving a practical speed for

real world imaging.

## Learned ADMM (Le-ADMM)

In the simplest of our unrolled networks, Le-ADMM (learned ADMM), we model each $k^{\text{th}}$ iteration of ADMM as a layer in a neural network, outlined in Fig. 4.3. In Le-ADMM, the optional denoiser step depicted in Fig. 4.3 is omitted. We denote the collection of update equations at the $k^{\text{th}}$ step of ADMM as $\mathbf{S}^k$. These update equations are given by:

$$\mathbf{S}^{k+1} \leftarrow \begin{cases} u^{k+1} \leftarrow \mathcal{T}_{\boxed{\tau^k}}(\mathbf{\Psi}(\mathbf{x}^k) + \alpha_2^k/\boxed{\mu_2^k}) & \text{sparsifying soft-thresholding} \\ v^{k+1} \leftarrow (\mathbf{C^T C} + \mu_1 I)^{-1}(\alpha_1^k + \boxed{\mu_1^k}\mathbf{H}\mathbf{x}^k + \mathbf{C^T b}) & \text{least-squares update} \\ w^{k+1} \leftarrow \max(\alpha_3^k/\boxed{\mu_3^k} + \boldsymbol{x}^k, 0) & \text{enforce non-negativity} \\ \mathbf{x}^{k+1} \leftarrow (\boxed{\mu_1^k}\mathbf{H^T H} + \boxed{\mu_2^k}\mathbf{\Psi^T \Psi} + \boxed{\mu_3^k} I)^{-1} r^k & \text{least-squares update} \\ \alpha_1^{k+1} \leftarrow \alpha_1^k + \boxed{\mu_1^k}(\mathbf{H}\mathbf{x}^{k+1} - v^{k+1}) & \text{dual for v} \\ \alpha_2^{k+1} \leftarrow \alpha_2^k + \boxed{\mu_2^k}(\mathbf{\Psi}(\mathbf{x}^{k+1}) - u^{k+1}) & \text{dual for u} \\ \alpha_3^{k+1} \leftarrow \alpha_3^k + \boxed{\mu_3^k}(\mathbf{x}^{k+1} - w^{k+1}) & \text{dual for w} \end{cases}$$

$$\text{where } r^k = ((\boxed{\mu_3^k} w^{k+1} - \alpha_3^k) + \mathbf{\Psi^T}(\boxed{\mu_2^k} u^{k+1} - \alpha_2^k) + \mathbf{H^T}(\boxed{\mu_1^k} v^{k+1} - \alpha_1^k)). \tag{4.4}$$

The trainable parameters are outlined in blue and can be summarized by $\Theta = \{\mu_1^k, \mu_2^k, \mu_3^k, \tau^k\}$, where $k$ represents the iteration number. For 5 unrolled layers, we have a total of 20 learned parameters. After a fixed number of ADMM iterations the reconstruction is compared to the ground truth (lensed) image using the loss function described in Section 4.4. The trainable parameters are updated using backpropagation to minimize this loss across multiple training examples. Le-ADMM can be interpreted as a data-tuned ADMM where the parameters that are typically hand-tuned or auto-tuned are now updated based on the data in order to minimize a data-driven loss function.

## Le-ADMM*, with learned regularizer

Le-ADMM* has the same overall structure as Le-ADMM, but also includes a learnable regularizer based on a CNN. The new update steps are summarized below:

$$\mathbf{S}^{k+1} \leftarrow \begin{cases} u^{k+1} \leftarrow \boxed{\boldsymbol{\mathcal{N}}}(\mathbf{x}^k) & \text{network regularizer} \\ v^{k+1} \leftarrow (\mathbf{C^T C} + \mu_1 I)^{-1}(\alpha_1^k + \boxed{\mu_1^k}\mathbf{H}\mathbf{x}^k + \mathbf{C^T b}) & \text{least-squares update} \\ w^{k+1} \leftarrow \max(\alpha_3^k / \boxed{\mu_3^k} + \boldsymbol{x}^k, 0) & \text{enforce non-negativity} \\ \mathbf{x}^{k+1} \leftarrow (\boxed{\mu_1^k}\mathbf{H^T H} + \boxed{\mu_2^k} I + \boxed{\mu_3^k} I)^{-1} r^k & \text{least-squares update} \\ \alpha_1^{k+1} \leftarrow \alpha_1^k + \boxed{\mu_1^k}(\mathbf{H}\mathbf{x}^{k+1} - v^{k+1}) & \text{dual for v} \\ \alpha_3^{k+1} \leftarrow \alpha_3^k + \boxed{\mu_3^k}(\mathbf{x}^{k+1} - w^{k+1}) & \text{dual for w} \end{cases} \tag{4.5}$$

$$\text{where } r^k = ((\boxed{\mu_3^k} w^{k+1} - \alpha_3^k) + \boxed{\mu_2^k} u^{k+1} + \mathbf{H^T}(\boxed{\mu_1^k} v^{k+1} - \alpha_1^k)).$$

$\boldsymbol{\mathcal{N}}$ represents a learnable network, applied at each ADMM iteration. For this learnable transform, we use a small U-Net based on [176] consisting of a single encoding and decoding step (complete architecture is available in Appendix A.1). Because $\boldsymbol{\mathcal{N}}$ does not represent the solution to a well-defined convex problem, we drop $\alpha_2$, the dual variable associated with $u$. The learnable parameters for Le-ADMM* are thus given by $\Theta = \{\mu_1^k, \mu_2^k, \mu_3^k, \boldsymbol{\mathcal{N}}\}$, which is a total of 32,135 learned parameters. The added learned parameters allow Le-ADMM* to learn a better prior on the data and account for model-mismatch errors in the forward model, at the price of requiring additional training data.

## Le-ADMM-U

Our third variation of unrolled networks is Le-ADMM followed by a learned denoiser, as shown in Fig. 4.3. Here, a U-Net is used as the denoiser [176]. This method has the most learnable parameters, having a total of 10,605,927 learned parameters, all but 20 of which are from the U-Net. The parameters of Le-ADMM-U, given by $\Theta = \{\mu_1^k, \mu_2^k, \mu_3^k, \tau^k, \boldsymbol{\mathcal{U}}\}$, are jointly updated throughout training. The Le-ADMM portion of the network performs the bulk of the deconvolution and includes knowledge of the forward model, while the U-Net denoises the final image, is able to correct model mismatch errors, and makes the images look more visually appealing. Our denoiser network architecture is described in Appendix A.1.

## U-Net

For completeness, we also compare to a purely deep method with no knowledge of the system physics in the reconstruction. For this, we directly use the U-Net architecture from [176], resulting in 10,605,907 learned parameters. We summarize this network architecture in Appendix A.1.

## Loss functions

The loss function must be carefully selected because it dictates the parameter updates throughout the training process. In classic methods, ground truth is unavailable, so the

loss is a function of the consistency of the final image, $\hat{\mathbf{x}}$, with the measurement model and any image priors. With the inclusion of ground truth training data pairs, we now have access to another class of loss functions that directly compare a given reconstructed image to its associated ground truth image, $\mathbf{x}_{gt}$. One common loss is the mean-squared error (MSE) loss with respect to the ground truth, $\|\mathbf{x}_{gt} - \hat{\mathbf{x}}\|_2^2$. However, MSE favors low frequencies and generally results in learned reconstructions that are blurry and lack detail [231]. Here, we will use the Learned Perceptual Image Patch Similarity metric (LPIPS) that uses deep features and aims to quantify a perceptual distance between two images, as introduced in [231]. During training, we use a combination of both MSE and LPIPS, as outlined in Section 4.5. These loss functions are summarized in Table 4.1.

**Table 4.1: Loss functions**

| | | |
|---|---|---|
| **Data Fidelity** | $\frac{1}{2}\|\mathbf{b} - \mathbf{CH}\hat{\mathbf{x}}\|_2^2$ | Consistency of the measurement with our knowledge of the imaging system |
| **MSE** | $\|\mathbf{x}_{gt} - \hat{\mathbf{x}}\|_2^2$ | Pixel-wise difference between reconstruction and ground truth |
| **LPIPS** | $\text{LPIPS}(\mathbf{x}_{gt}, \hat{\mathbf{x}})$ | Perceptual distance between reconstruction and ground truth [231] |

## 4.5 Implementation

For training, we simultaneously collect a set of lensless and ground truth image pairs using an experimental setup consisting of a lensed camera, a DiffuserCam, a beamsplitter, and computer monitor (Fig. 4.1). The cameras and computer monitor are simultaneously triggered, which allows us to display and capture all the training pairs in the dataset overnight.

Our DiffuserCam prototype consists of an off-the-shelf diffuser (Luminit 0.5°) with a laser-cut paper aperture placed approximately 9 mm from a CMOS sensor. The lensed camera is focused at the plane of the computer screen, approximately 10 cm away. We capture a calibration PSF (see Fig. 4.1) using an LED point source placed at the distance of the computer screen, which sets the focal plane of the DiffuserCam. For both DiffuserCam and the ground truth camera, we use Basler Dart (daA1920-30uc) sensors. We use a 6 mm S-mount lens for the ground truth camera and calibrate the lens distortion using OpenCV's undistort camera calibration procedure [29]. To achieve pixel-wise alignment between the image pairs, we first optically align the two cameras, then further calibrate by displaying a series of points on the computer monitor that span the field-of-view. We reconstruct these point images and compute the homography transform needed to co-align both cameras' coordinate systems. This transform is applied to all subsequent images.

Our dataset consists of 25,000 images from the MirFlickr dataset [94]. The raw data from each sensor is 1920×1080 pixels, but is down-sampled by a factor of 4 in each direction, to 480×270. This is necessary due to moiré fringes from the screen which degrade our lensed

image quality. We split the dataset up into 24,000 training images and 1,000 test images. Our networks are implemented in PyTorch and trained on a Titan X GPU, using an ADAM optimizer throughout training [106]. We find that using a combined loss based on MSE and LPIPS works best in practice. We weight MSE more heavily during earlier epochs and weight LPIPS more heavily during later epochs for further refinement. Source code is available at [157]. When displaying the final images, we crop to 380×210 pixels to avoid displaying areas beyond the borders of the computer monitor.

## 4.6 Results

After training, we compare the performance of our unrolled networks against both classic ADMM and the fully deep U-Net. Since the number of iterations of ADMM affects both speed and quality of the result, we compare against both ADMM run until convergence (100 iterations) as well as ADMM bounded to 5 iterations. Bounded ADMM takes a similar time to run as our unrolled networks and converged ADMM sets a baseline for the best performance classic algorithms can achieve. On the deep side, we compare against a U-Net which is trained using our raw DiffuserCam measurements and ground truth labels.

The reconstruction results of images in our test set (taken by the monitor setup, but not used during training) show that our fastest learned networks are able to produce similar or better images than converged ADMM in the same amount of time as bounded ADMM (5 iterations), a 20× speedup while achieving comparable or better image quality. Furthermore, we show reconstructions of natural images in the wild (not from a computer monitor), demonstrating that our networks are able to generalize to 3D objects with variable lighting conditions.

**Table 4.2: Network performance on test set**

| Reconstruction | Data Fidelity | MSE | LPIPS | Time on GPU (ms) | # Training Images |
|---|---|---|---|---|---|
| ADMM (converged) | 13.62 | .0622 | .5711 | 1,520 | none |
| ADMM (bounded) | 11.32 | .1041 | .6309 | 71 | none |
| **Le-ADMM** | 13.70 | .0618 | .4434 | 71 | 100 |
| **Le-ADMM*** | 16.21 | .0309 | .327 | 200 | 100 |
| **Le-ADMM-U** | 22.14 | **.0074** | **.1904** | 75 | 23,000 |
| U-Net | 19 | .0154 | .2461 | 10 | 23,000 |

### Test set results

Table 4.2 summarizes the reconstruction performance and speed of our learned networks on the test set. Here we can see that our fastest networks (Le-ADMM and Le-ADMM-U) are


Figure 4.4: **Test set results**, with the raw DiffuserCam measurement (contrast stretched) and the ground truth images from the lensed camera for reference. Le-ADMM (71 ms) has similar image quality to converged ADMM (1.5 s) and better image quality than bounded ADMM (71 ms). Le-ADMM* and Le-ADMM-U have noticeably better visual image quality. The U-Net by itself is unable to reconstruct the appropriate colors and lacks detail.

20× faster than classic reconstruction algorithms (ADMM converged) and have similar or better average MSE and LPIPS scores. Le-ADMM* is slightly slower due to its inclusion of a CNN on the uncropped image in each unrolled layer, however is still an order of magnitude faster than converged ADMM. As we move on the scale from classic to deep (Le-ADMM → Le-ADMM* → Le-ADMM-U), our networks have better MSE and LPIPS scores, but have worse data fidelity.

Figure 4.4 shows several sample images from our test set reconstructions. Here we can see that our networks (Le-ADMM, Le-ADMM*, Le-ADMM-U) produce images that are of equal or better quality than converged ADMM. We can see that bounded ADMM has streaky artifacts, but our learned networks do not. Le-ADMM-U has the best reconstruction performance overall and produces images that are visually similar to the ground truth images. Overall, Le-ADMM-U has 3× better image quality than converged ADMM as measured by

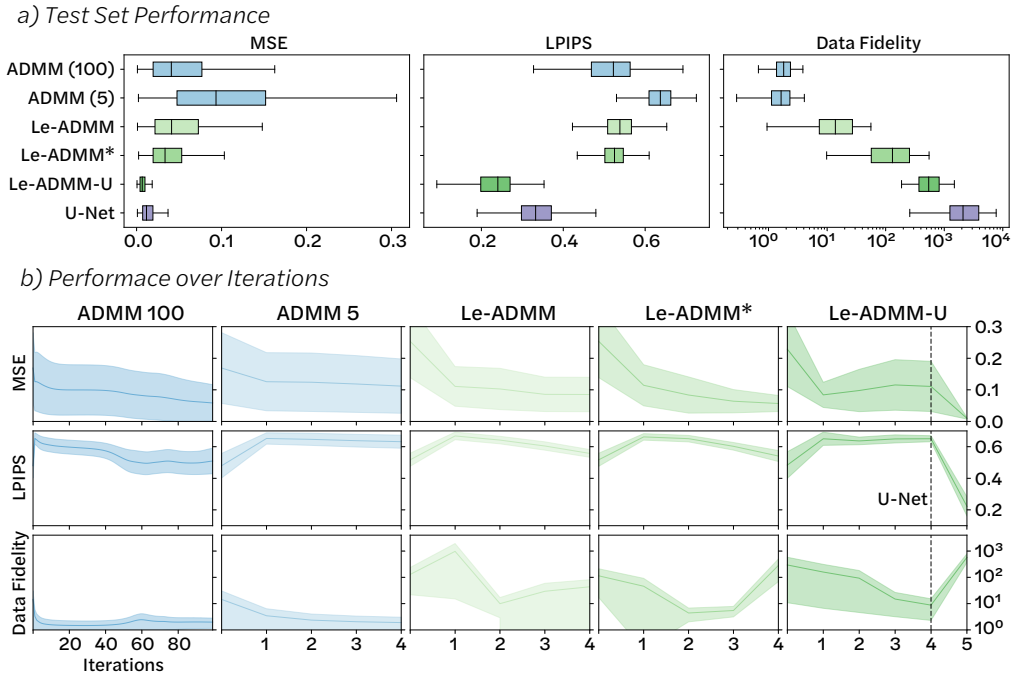


Figure 4.5: **Network performance on test set**. (a) Here we plot the MSE, LPIPS, and Data Fidelity values for all image pairs in our test set. On average, our learned networks (green) are more similar to the ground truth lensed images (lower MSE and LPIPS) than 5 iterations of ADMM. Furthermore, our networks have comparable performance to ADMM (100), which takes 20× longer than Le-ADMM and Le-ADMM-U. However, the data fidelity term is higher for the learned methods, indicating that these reconstructions are less consistent with the image formation model. (b) Here we plot performance after each layer (or equivalently, each ADMM iteration) in our network, showing that MSE and LPIPS generally decrease throughout the layers. The U-Net denoiser layer in Le-ADMM-U significantly decreases the LPIPS and MSE values, at the cost of data fidelity.

the LPIPS metric. The U-Net does not perform as well as Le-ADMM-U, having inconsistent colors and missing higher frequencies. This shows the utility in combining model-based and deep methods.

Figure 4.5(a), plots the distribution of MSE, LPIPS, and Data Fidelity scores for the test set. We can see that Le-ADMM-U has the best LPIPS and MSE scores and outperforms converged ADMM, whereas Le-ADMM has similar LPIPS and MSE scores to converged ADMM with many fewer training pairs. Here we can clearly see the trend of data fidelity increasing as MSE and LPIPS decrease, showing that there is a trade-off between image quality and matching the imaging model. We interpret this as our system model being imperfect, which prevents purely model-based algorithms from achieving the best image quality. As we increase the number of learned parameters, we are able to correct artifacts introduced by model mismatch, producing more visually appealing images that better match

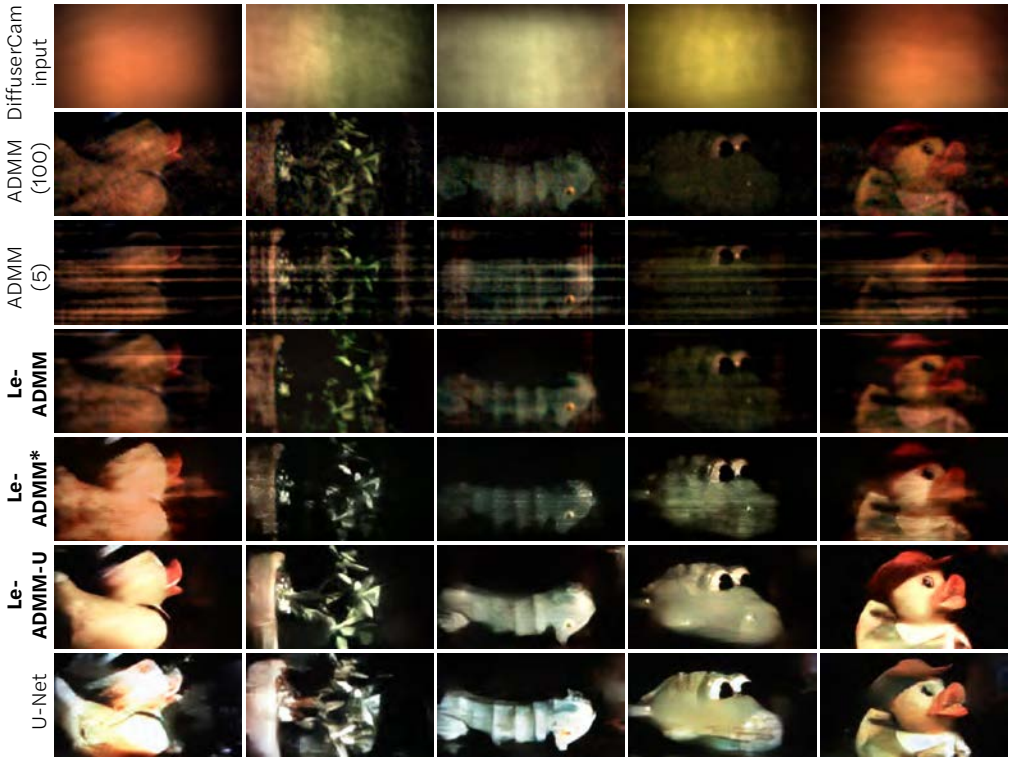


Figure 4.6: **Network performance on objects in the wild** (toys and a plant) captured with our lensless camera. We show the raw measurement (contrast stretched) on the top row, followed by converged ADMM, ADMM bounded to 5 iterations, our learned networks, and U-Net for comparison. Our learned networks have similar or better image quality as converged ADMM, and Le-ADMM-U has the best image quality. For instance, Le-ADMM-U is able to capture the details in the sideways plant (second column from left) and the eye of the toy duck (right). The U-Net alone has good image quality, but is missing some colors and details (e.g. the first image is washed out and the nose of the alligator toy is miscolored).

the lensed camera. Figure 4.5(b) analyzes what happens to the reconstruction throughout the layers of the learned network. The MSE and LPIPS scores tend to decrease with each iteration, while data fidelity increases. For Le-ADMM-U, the U-Net greatly improves the LPIPS and MSE values at the cost of data fidelity.

## Generalization to images in the wild

Next, we remove the computer monitor and capture DiffuserCam images of natural objects. Figure 4.6 shows some example reconstructions using our learned networks. Again, we see that our networks produce images of similar or higher visual quality than converged ADMM. In particular, Le-ADMM-U again produces the most visually appealing images and

has better image quality than converged ADMM. This shows that our learned networks are able to generalize beyond imaging a computer monitor to situations with dramatically different lighting conditions.

## 4.7 Discussion

Our work presents a preliminary analysis of using unrolled, model-based neural networks on a real experimental lensless imaging system. We show that it is favorable to choose a network that combines classic and deep methods. We can perform comparably to classic algorithms at a fraction of the speed using only a few learned parameters, and we can greatly improve image quality when increasing the number of learned parameters. In addition, the number of learned parameters in the network could be varied depending on the application. For instance, scientific imaging applications might choose to have fewer learned parameters to prevent overfitting to the training data. Meanwhile, photography applications may prefer a deeper method with more parameters, potentially producing more visually appealing images at the expense of possibly hallucinating details not present in the scene.

The quality and resolution of our reconstructions is bounded by that of our training dataset, including any potential imperfections in the physical system. For instance, any aberrations introduced by our lensed camera or beamsplitter will affect the learned reconstructions, since the lensed images are used as the ground truth when updating the network parameters. However, in practice we correct for aberrations such as distortion before training; other effects (e.g. chromatic aberration, field curvature) are negligible at our reconstruction grid size. Possible future work includes training on scenes with larger depth content to yield reconstructions with desirable defocus blurs, such as seen in a lensed camera.

## 4.8 Conclusion

We presented several unrolled, model-based neural networks for lensless imaging with a varying number of trainable parameters. Our networks jointly incorporate the physics of the imaging model as well as learned parameters in order to use both the known physics and the power of deep learning. We presented an experimental system with a prototype lensless camera that was used to rapidly acquire a dataset of aligned lensless and lensed images for training. Each of our networks are able to produce similar or better image quality compared to standard algorithms, with the fastest offering a 20× improvement. In addition, our deeper method, Le-ADMM-U has 3× better image quality than standard algorithms on the LPIPS perceptual similarity scale. Our learned network is fast enough for interactive previewing of the scene and also produces visually appealing images, addressing two of the big limitations of lensless imagers. Our work suggests that using such model-based neural networks could greatly improve imaging speed and quality for lensless imaging at the cost of a training step before camera operation.

# Chapter 5

# Physics-informed Learning for Single-shot 3D Imaging with Spatially-varying Deconvolution

In this chapter[1], we introduce a physics-informed reconstruction network for spatially-varying deconvolution. Spatially-varying aberrations complicate imaging inverse problems since convolutional models do not hold for spatially-varying systems, making reconstruction algorithms more computationally intensive. Unfortunately, many real microscopes have spatially-varying aberrations. We tackle the problem of spatially-varying deconvolution by introducing a physics-informed network that first approximately inverts the effects of the spatially-varying blur using multiple differentiable Wiener filters, then refines the estimate using a CNN. This is an example of using an approximate physics-based inversion followed by a CNN for refinement. We train our physics-informed network using entirely simulated data, then validate the performance on experimental data. Overall, we demonstrate a 625× speedup for 2D deconvolutions and a 1600× speedup for 3D deconvolutions compared to classic methods, while providing better performance than other deep-learning-based methods that do not account for spatially-varying blur.

Deconvolution is integral to many modern imaging systems. Imperfections in the optics may inadvertently blur the image (e.g. aberrations) and deconvolution can be used to computationally undo some of this blur [186, 179]. In microscopy, deconvolution can reduce out-of-focus fluorescence to provide sharper 3D images [149, 22, 182]. Alternatively, distributed point spread functions (PSFs) can be intentionally designed into an imaging system in order to enable new capabilities, such as single-shot 3D [223, 114, 131, 8, 15] or hyperspectral imaging [158, 95]. In this case, multiplexing optics encode 2D or 3D information by mapping each point in object space to a distributed pattern on the image sensor, then deconvolution is used to recover the encoded image or volume. In either case, a deconvolution algorithm is needed in order to recover a clear image or volume from the blurred or encoded

[1]This chapter is based on the published journal paper titled "Deep learning for fast spatially varying deconvolution" and is joint work with Kyrollos Yanny, Richard Shuai, and Laura Waller [221].

measurement.

A variety of algorithms have been utilized for deconvolution over the years. Classical methods range from closed-form approaches such as Wiener filtering to iterative optimization approaches, such as Richardson-Lucy and the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA). Many methods incorporate hand-picked priors, such as Total Variation (TV) and native sparsity, to improve image quality. These approaches often assume that the system is shift-invariant, meaning that all parts of the image have the same blur kernel. Shift-invariance allows the forward model to be efficiently expressed as a convolution between the PSF and the object. However, most imaging systems will have a blur that varies across the field-of-view (FoV)—that is, they have spatially-varying PSFs, usually due to field-varying aberrations. This motivates the use of spatially-varying deconvolution, for which several methods have been proposed [10, 167, 141, 21, 54, 223, 114]. Unfortunately, many of these algorithms are prohibitively slow and computationally intensive, making them unsuitable for real-time image reconstruction. Furthermore, these methods can suffer from poor image quality, especially for highly multiplexed imaging systems that have PSFs with large spatial extent, or for poorly chosen priors. Recently, deep-learning based deconvolution methods have been demonstrated to improve both image quality and reconstruction speed, providing a promising improvement over iterative approaches [156, 197, 176, 105]. However, to date, these methods rely on a shift-invariant PSF approximation and do not generalize well to optical systems with field-varying aberrations.

## 5.1 Method Overview

In this work, we propose a new deep-learning based approach for fast, spatially-varying deconvolution. Our network, termed MultiWienerNet, consists of multiple learnable Wiener deconvolutions followed by a refinement convolutional neural network (CNN). The Wiener deconvolution layer performs multiple Fourier-space deconvolutions, each with a different PSF from a particular field point, yielding several intermediate images which have sharp features in different regions of the image. These intermediate images are then fed into the refinement CNN which fuses and refines them to create the final sharp deconvolved image. The learnable Wiener deconvolution filters are initialized with PSFs captured at several locations in the FoV, but then allowed to update throughout training to learn the best filters and noise regularization parameters. This allows us to incorporate knowledge of the field-varying aberrations into the network, providing a physically-informed initialization that is further refined throughout training. The end result is a fast spatially-varying deconvolution that is $625-1600\times$ faster than the baseline iterative method (Spatially-Varying FISTA [223]), enabling real-time image reconstruction. In addition, incorporating the field-varying PSFs allows our network to have better image quality near the edges of the FoV than is achieved by existing deep learning based methods which assume shift-invariance.

Our approach consists of the following steps: 1) generating a simulated training dataset by applying measured PSFs to images from open-source microscopy datasets, 2) initializing and training the MultiWienerNet using the simulated data, and 3) utilizing the trained

network for fast shift-varying deconvolutions, where the input to the network is a single measurement. To demonstrate, we choose the challenging example of single-shot 3D microscopy with Miniscope3D [223] as our test case. Miniscope3D uses a phase mask that consists of a random array of multi-focal microlenses to encode 3D information in a 2D image. The system maps each object point in the FoV to a unique pseudorandom pattern on the sensor, then decodes the captured images by solving a sparsity-constrained inverse problem. We select this system both for its spatially and depth-varying PSFs, Fig. 5.1, and its high degree of multiplexing, which creates a particularly challenging deconvolution problem. We demonstrate our approach on both 2D deconvolution, where the goal is to recover a 2D image from a 2D measurement, as well as 3D deconvolution, where the goal is to recover a 3D volume from a single 2D measurement.

## 5.2 Spatially-varying Forward Model

To generate simulated datasets, we first need a forward model that can faithfully relate how a 3D object is mapped to a 2D measurement in our microscope system, taking into account the effects of spatially-varying blur introduced by the system. To establish this forward model, the volumetric object intensity is treated as a 3D grid of voxels, $\mathbf{v}[x, y, z]$. Each voxel produces a PSF, $\mathbf{h}[x', y'; x, y, z]$, on the camera sensor, where $[x', y']$ are image space indices. Since the object voxels are mutually incoherent, the measurement can be expressed as a linear combination of the PSFs from each voxel in the object:

$$\begin{aligned}\mathbf{b}[x', y'] &= \sum_z \sum_{x,y} \mathbf{v}[x, y, z]\mathbf{h}[x', y'; x, y, z] \\ &= \mathbf{A}\mathbf{v},\end{aligned} \tag{5.1}$$

where $\mathbf{b}$ is the measurement and $\mathbf{A}$ is a matrix that maps the 3D volume to the 2D measurement. For the shift-invariant case—where the PSF is the same at all points within the FoV for each depth—Eq. 5.1 reduces to a sum over 2D convolutions:

$$\mathbf{b}[x', y'] = \sum_z \sum_{x,y} \mathbf{v}[x, y, z] \overset{[x,y]}{*} \mathbf{h}[x, y, z], \tag{5.2}$$

where $\overset{[x,y]}{*}$ represents a 2D convolution. However, this shift-invariant assumption is generally not true and its corresponding convolutional forward model will lead to inaccurate deconvolution. There are multiple ways to approximate the image formation model for shift-invariant PSFs (e.g. locally convolutional, low-rank models, etc. [10, 141, 21, 54, 223, 66, 114]). Any of these techniques is applicable to our pipeline. In particular, we choose to use the low-rank model from [223], approximating the spatially-varying PSFs as a weighted sum of shift-invariant kernels:

$$\mathbf{b}[x', y'] = \sum_z \sum_{r=1}^{K} \left\{ (\mathbf{v}[x, y, z]\mathbf{w}_r[x, y, z]) \overset{[x,y]}{*} \mathbf{g}_r[x, y, z] \right\} [x', y'], \tag{5.3}$$

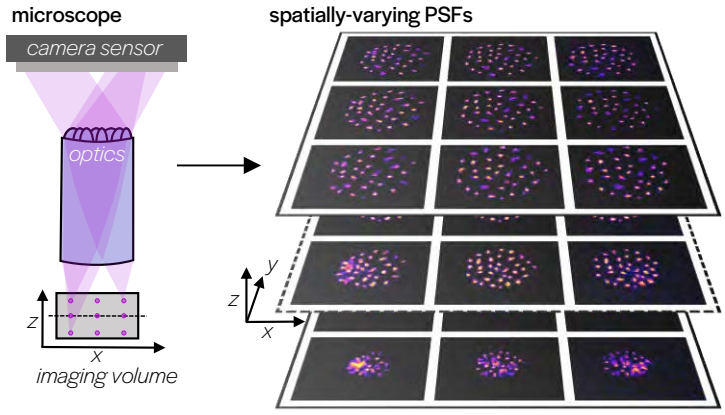


Figure 5.1: **Spatially-varying Point Spread Functions (PSFs)**. (left) Simplified diagram of Miniscope3D showing a lateral and axial scan of a point source through the volumetric field-of-view (FoV) to capture the spatially-varying PSFs. (right) Experimental images of the PSFs from different points in the volumetric FoV, which are used to initialize the Wiener deconvolution layer of our MultiWienerNet method.

where the weights $\{\mathbf{w}_r\}$ and the kernels $\{\mathbf{g}_r\}$ are computed from a singular value decomposition (SVD) of sparsely sampled PSFs from different positions in the FoV and the inner sum is over the $K$ largest values in the SVD. Note that in the 2D imaging case, the object is a thin slice in the $z$-dimension, so the outer sum over $z$ is not included in the forward model.

## 5.3 Simulating a Realistic Dataset

Using this forward model, we can simulate measurements from our microscope to use in training datasets. We run images from online microscopy datasets [109, 12, 136, 234] through the low-rank forward model (see below), generating pairs of ground truth volumes/images and simulated measurements. For the 3D dataset, the CytoPacq [40] simulator is particularly helpful in obtaining time-series 3D volumes.

Given a good system forward model, it is possible to generate any number of image pairs, which we can use to train our MultiWienerNet. We generate both 2D and 3D training datasets; the 2D dataset contains 2D target objects with dimensions (x,y,z) of (336,480,1), representing a FoV of $700 \times 1000\ \mu m^2$, while the 3D datasets contain 3D target objects with dimensions (x,y,z) of (336,480,32), representing a FoV of $700 \times 1000 \times 320\ \mu m^3$. We generate $5,000$ 2D and $15,000$ 3D training images, with an 80/20 training/testing split.

### System Calibration

A simple calibration procedure based on scanning a fluorescent bead (point source) across the field-of-view (FoV) can be used to characterize the spatially-varying behavior of the system.

Rather than calibrating the PSF for every possible field point, we adopt the calibration method presented in [223], and sparsely sample the PSFs across the field, then use a low-rank forward model to account for the shift-variance. In this work, we sample the PSF at 64 locations across the FoV for each depth. This can be done by simply scanning a bead and taking 64 calibration images across an 8×8 grid. If the bead is particularly dim, or there is noticeable noise present in the image, averaging multiple images together can help reduce noise and provide a better calibration image. If 3D deconvolution is desired, this calibration must be repeated for each depth of interest. In our case, we repeat this procedure for 32 different depths within the depth range of the microscope. To save time on calibration, it is also possible to do this calibration using a sample of unstructured beads instead of a single calibration bead [115].

### Simulating measurements

The following pre-processing steps are used to generate our simulated measurements:

1. Resize the image/volume to fit the Miniscope3D image/volume size which is $336 \times 480 \times 1$ for 2D reconstruction and $336 \times 480 \times 32$ for 3D reconstruction
2. Using the sparsely sampled calibration PSFs, compute the weights $\{\mathbf{w}_r\}$, and the kernels $\{\mathbf{g}_r\}$, from a singular value decomposition (SVD) procedure outlined in [223].
3. Simulate measurements by running the resized data through the low-rank forward model in Eq. 3 of main text
4. Add appropriate levels of Gaussian and Poisson noise to the simulated measurement.

These pre-processing steps can be adapted to a new system by altering the desired image/volume size and using the calibrated PSFs for the new system.

## 5.4 Physics-informed Network Architecture

Our network consists of two components: a differentiable Wiener deconvolution layer, and a refinement CNN, Fig. 5.2. Wiener deconvolution is a fast and simple approach that is used for linear shift-invariant systems given a known PSF and noise level. It consists of a single Fourier filtering step, which can be efficiently computed using FFTs. However, when the assumption of shift-invariance does not hold, Wiener deconvolution results in degraded image quality in the areas of the image in which the PSF differs from the one assumed. Hence, instead of performing Wiener deconvolution with a single PSF [105], we approximate the behavior of our spatially-varying system using $M$ PSFs taken from different field points. Our Wiener deconvolution layer thus performs $M$ Wiener deconvolutions, resulting in $M$ intermediate deconvolved images, as shown in Fig. 5.2. Each will have sharp features in a different region of the image, corresponding to the area in the FoV from which the PSF was

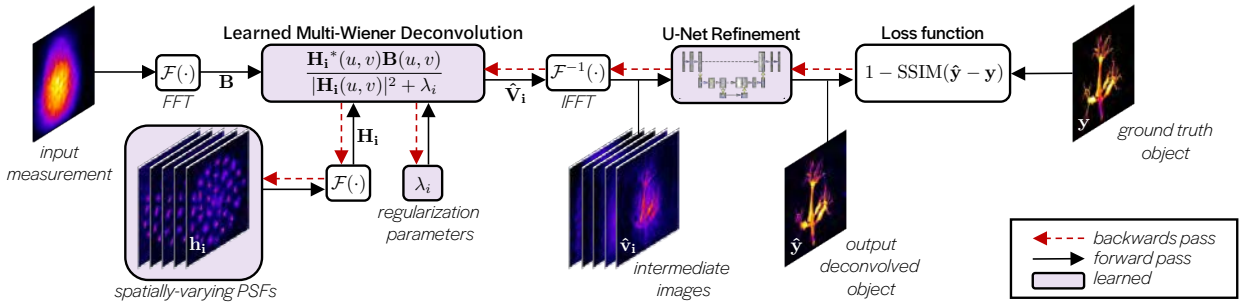


Figure 5.2: **MultiWienerNet architecture**. Our pipeline consists of two parts: 1) a learnable multi-Wiener deconvolution layer which is initialized with knowledge of the system's spatially-varying PSFs and outputs a set of deconvolved intermediate images. 2) A U-Net refinement step which combines and refines the intermediate images into a single output image. Both parts are jointly-optimized during training using simulated data. After training, experimental measurements are fed into the optimized MultiWienerNet for fast spatially-varying deconvolution.

taken. These $M$ intermediate images are then fed into the refinement CNN which combines and refines the images to produce the final image/volume.

Mathematically, our differentiable Wiener deconvolution layer can be described as follows:

$$\hat{\mathbf{V}}_i(u,v) = \frac{\mathbf{H}_i^*(u,v)\mathbf{B}(u,v)}{|\mathbf{H}_i(u,v)|^2 + \lambda_i}, \quad i = 1, 2, ..., M, \tag{5.4}$$

where $(u, v)$ are frequency-space coordinates, $\mathbf{B}(u, v)$ is the Fourier transform of the measurement, $\hat{\mathbf{V}}_i(u, v)$ is the $i$th Fourier transform of the estimated scene intensity, $\mathbf{H}_i(u, v)$ is the Fourier transform of the $i$th PSF, $^*$ denotes a complex conjugate and $\lambda_i$ is a regularization parameter related to the signal-to-noise ratio (SNR) of the measurement. Note that the intermediate images are obtained after taking an inverse Fourier transform, $\hat{\mathbf{v}}_i = \mathcal{F}^{-1}(\hat{\mathbf{V}}_i)$. Here, the $\mathbf{H}_i(u, v)$ are initialized using the $M$ PSFs measured from different points in the FoV. For 3D deconvolution, PSFs are sampled at multiple depth planes. Both $\mathbf{H}_i(u, v)$ and $\lambda_i$ are learnable and optimized during training. Finally, the $M$ intermediate images are fed into the U-Net refinement step which consists of a 2D U-Net for the 2D deconvolution problem or a 3D U-Net for the 3D problem [176].

For 2D deconvolution, $\mathbf{H}_i$ is initialized with measured PSFs from field points sampled on a $3 \times 3$ grid across the FoV, giving $M = 9$. For 3D, the PSFs are sampled on a $3 \times 3 \times 32$ grid across the FoV, giving $M = 288$. After the model is initialized with the $M$ measured PSFs, the simulated pairs of measurements and ground truth volumes/images are used in training to update the parameters of the MultiWienerNet, including $\mathbf{H}_i$, $\lambda_i$, and all the parameters in the U-Net. We use a structural similarity index measure (SSIM) loss and L1 loss, which generally outperforms mean squared error (MSE) or L1 loss on its own.

### Model initialization and network architecture

Before training, the MultiWienerNet must be initialized with the spatially-varying PSFs, $\mathbf{h}_i$, and additionally with regularization parameters, $\lambda_i$. Here, we utilize 9 PSFs from a 3×3 grid from each depth plane in our FoV to initialize $\mathbf{h}_i$. The number of initialized filters can be updated based on the level of spatial-variance of the system. Given more spatial-variance, a larger number of filters should be used; however, this comes at the price of added computational complexity. We found that 9 filters was sufficient for the Miniscope3D. For the U-Net architecture, our contracting path consists of four repeated applications of two $3\times3$ convolutions, followed by a Scaled Exponential Linear Unit (SELU) and a convolutional down-sampling layer with stride 2. The expansive path consists of four repeated applications of two $3 \times 3$ convolutions, followed by a SELU and a transposed convolution up-sampling layer with strides 2.

### Training and Testing Details

We train for 25 epochs for the 3D reconstruction case and 100 epochs for the 2D case, using a learning rate of $10^{-4}$. We use the ADAM optimizer [106] with default parameters throughout training. We allow the Wiener deconvolution filters and regularization parameters to update throughout training. We tested having the Wiener filters and/or the regularization parameters be fixed throughout training, but the best results were obtained when they were allowed to change. Our loss function is an average of a structural similarity index measure (SSIM) loss and L1 loss. For SSIM, we use an $11 \times 11$ Gaussian filter of width 1.5. We test the performance on both simulated and experimental data. The experimental sample in main-text Figure 3(a) is a 1951 USAF fluorescent resolution target, and in 3(b) is a freely moving stained tardigrade (water bear) captured at 40 frames per second.

## 5.5 Results and Analysis

After training, we test our model on $1,000$ images in a held-out test set from the online datasets, and on experimental data from the Miniscope3D setup. We compare our results against iterative spatially-varying FISTA, a U-Net, and the U-Net with a single Wiener deconvolution [105]. Our results show that the MultiWienerNet achieves more than 625× speedup in 2D reconstruction and 1600× speedup in 3D reconstruction as compared to FISTA, while also providing better PSNR and image quality, especially towards the edges of the FoV, where off-axis aberrations dominate. Our network also outperforms current deep-learning approaches while only being slightly slower. In addition, despite being trained solely on simulated data, the MultiWienerNet generalizes well to experimental data. Though FISTA achieves slightly higher resolution near the center of the FoV (Fig. 5.3(a)), MultiWienerNet performs better overall.

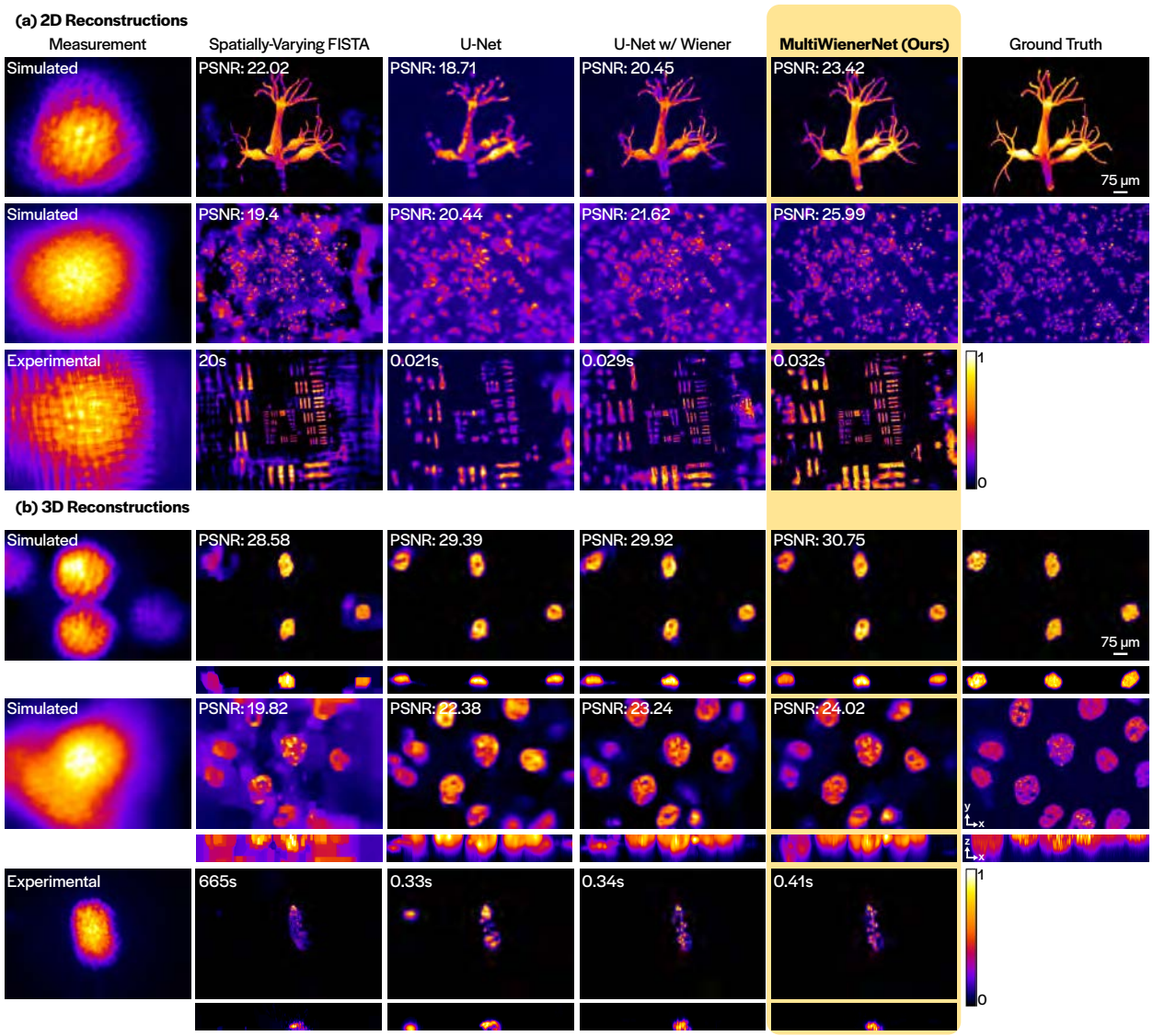


Figure 5.3: **Simulation and Experimental Results.** Deconvolution results for (a) 2D and (b) 3D, showing both simulated and experimental data. MultiWienerNet achieves better performance than other deep learning-based approaches that do not incorporate knowledge of a spatially-varying PSF (U-Net and U-Net w/Wiener), and achieves better and faster ($625-1600\times$ speedup) results than spatially-varying FISTA, which has poor reconstruction quality at the edges of the FoV, where spatially-varying aberrations are severe. For the 3D results, $xy$ and $xz$ maximum projections are shown.

## Timing results

We compare our reconstruction time against FISTA with a spatially-varying model and existing deep learning based models (U-Net, U-Net w/Wiener) by averaging over 100 runs

on GPU. The results are summarized in Table 5.1. We can see that our method is significantly faster than existing spatially-varying methods, and has comparable speeds to existing deep learning based methods. For 2D, we can perform reconstructions at 30 fps for 336×480 images. For 3D, we can perform reconstructions at 2.4 fps for 336×480×32 volumes. This is significantly faster than what is possible without using deep learning-based methods (0.05 fps for 2D, 0.0015 fps for 3D), and could enable interactive previewing. Timing results were performed on an RTX 2080 Ti GPU. The stopping criterion for FISTA was chosen to avoid unnecessary iterations. FISTA terminates if the solution is stable within a certain threshold, the loss is below a certain threshold, or the number of iterations exceeds a certain threshold.

Table 5.1: Reconstruction timing comparisons (GPU)

| | FISTA | U-Net | WienerNet | **MultiWeinerNet** |
|---|---|---|---|---|
| 2D | 20s | 0.021s | 0.029s | 0.032s |
| 3D | 665s | 0.33s | 0.34s | 0.41s |

### Learned filters

We allow the network to learn the Wiener deconvolution filters. We initialize the network with a grid of measured filters, $3 \times 3$ for 2D and $3 \times 3 \times 32$ for 3D, each is centered on a different region in the FoV. Fig. 5.4 shows initial filters at different positions demonstrating the shift-variance of the system. By allowing the network to update the filters, the network can find filters that better approximate the shift-variance or better recover certain frequencies. Fig. 5.5, shows the initialized and learned filters. The learned filters maintain similar structure in the central region as the initialized ones, while adding more information towards the edges of the filter. At first glance, it is unclear if the extra information added to the learned filters is useful. However, by examining the intermediate result of the Wiener deconvolution step using both the initialized and learned filter in Fig. 5.6, we find that the deconvolved image using the learned filter has much sharper features than the one using the initialized filter. This allows the multiple learnable Wiener deconvolutions to provide the CNN with sharp intermediate images that the CNN can further refine to remove low frequency artifacts. Note that we do not require the learned filters to be positive.

## 5.6 Conclusions

In summary, we propose a new network architecture to perform fast deconvolution for microscopes with spatially-varying PSFs. Given knowledge of the system's spatially-varying PSFs, our proposed network is trained in simulation, fusing known system parameters in the form of multiple differentiable Wiener deconvolutions with a CNN refinement step. After training, our network provides a $625 - 1600\times$ speedup over existing spatially-varying deconvolution algorithms and improved reconstruction quality, especially at the edges of the

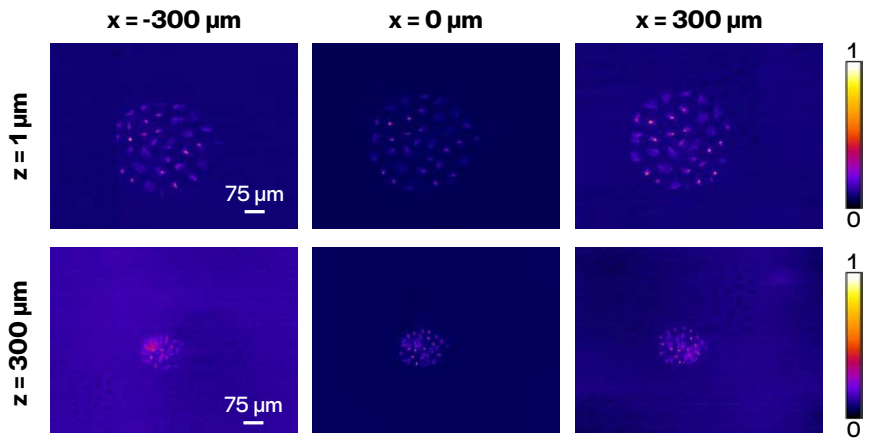


Figure 5.4: **Spatially varying PSFs**. Measured PSFs at different lateral positions for two different depth planes. Due to field-varying aberrations in the system, the PSFs change structure across the field-of-view (FoV). Note that the images are contrast stretched to show detail.

FoV. The code is open-source and can be utilized for imaging system with spatially-varying aberrations.

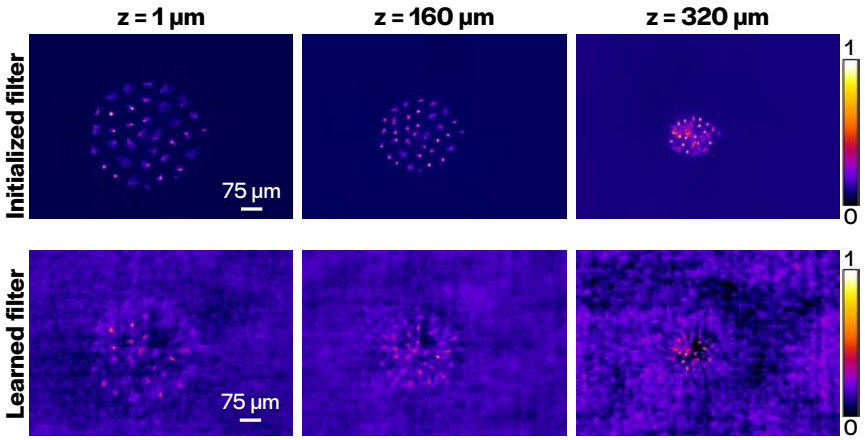


Figure 5.5: **Learned and initialized filters**. Central learned and initialized filters at different depths used for the 3D reconstruction. For visualization, we scale the learned filters from 0 to 1. In general, the filters can contain negative values. Note that the images are contrast stretched.

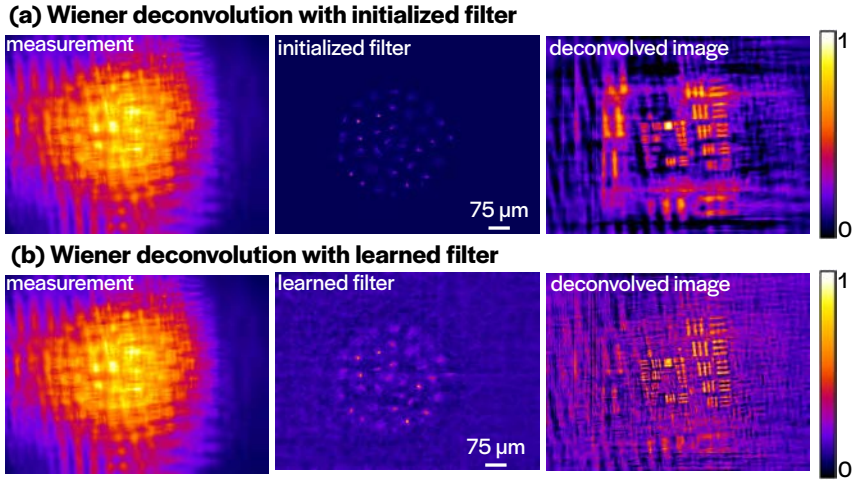


Figure 5.6: **Wiener deconvolution with learned filters**. (a) Using the central *initial* filter only, with a shift-invariance assumption. (b) Using the central *learned* filter with a shift invariance assumption results in sharper features in the deconvolved image.

# Chapter 6

# Unsupervised Learning for Compressive Lensless Photography

In this chapter[1], we introduce a physics-informed unsupervised method for the reconstruction of compressive, single-shot hyperspectral and video imaging. For many computational cameras, obtaining a large dataset of image pairs for training can be challenging. In this work, we show how we can use unsupervised machine learning to improve reconstruction performance, without the need for any training data.

Compressive imagers aim to recover more samples than they measure by leveraging compressive sensing, which guarantees signal recovery for underdetermined systems given certain assumptions about signal sparsity as well as incoherence in the measurement domain [35, 37]. For optical imaging, compressive imagers have been demonstrated for many applications, including single-pixel and coded-aperture cameras [59, 210, 92]. These imagers have been shown to be effective for a number of compressive sensing tasks, such as single-shot lightfield imaging [146], 3D imaging [123], hyperspectral [209, 72], and high-speed video imaging [90, 173, 137, 70]. Recently, there has been a push to make compressive imagers more accessible by using inexpensive and compact hardware [95, 135]. One such promising method includes lensless, mask-based imagers, which remove the lens of a traditional camera and instead place a phase or amplitude mask near the sensor to randomize the measurement, approximately achieving the measurement domain incoherence required for compressive sensing. These lensless mask-based cameras can be incredibly compact, with the mask only millimeters from the sensor, and assembly does not require precise alignment [15, 8, 185]. Mask-based lensless imagers have been demonstrated for compact single-shot 3D fluorescence microscopy [4, 114, 223, 131], hyperspectral imaging [158], and high speed single-shot video [9]. These sorts of cameras are very promising for a number of imaging modalities; however, their performance depends on the fidelity of the reconstruction algorithm. Since the algorithm must solve an underdetermined inverse problem with assumptions on signal sparsity, the choice of algorithm and imaging priors used can significantly impact results.

[1]This chapter is based on the published journal paper titled "Untrained networks for compressive lensless photography" and is joint work with Vi Tran, Grace Kuo, and Laura Waller [159].

Traditionally, images from compressive cameras are recovered by solving a convex optimization problem, minimizing both a least-squares loss based on the physics of the imaging system and a hand-chosen prior term, which enforces sparsity in some domain, Fig. 6.1(a). For successful image recovery by compressive sensing, there must be both incoherence in the sensing basis and sufficient sparsity in the sample domain. Lensless mask-based cameras utilize multiplexing optics to fulfill the incoherent sampling requirement, mapping each pixel in the scene to many sensor pixels in a pseudo-random way. The prior term enforces sparsity and ensures successful signal recovery. Over the years, a number of different hand-picked priors have been used for compressive imaging, such as sparsity in wavelets, total-variation (TV), and learned dictionaries. TV, which is based on gradient sparsity, has been particularly popular [126]. These methods are effective at solving imaging inverse problems, however they have recently been outperformed by deep learning-based methods, which incorporate non-linearity and network structures that may be better suited to image representation [97]. Recent work on plug and play (PnP) priors [178, 207, 230, 175] offers some hope of combining inverse problems with state of the art denoisers (both deep and not, e.g. BM3D [51]); however, PnP still relies on using hand-picked denoisers or pre-trained networks that may not be well-suited for a given application.

Recently, several deep-learning based approaches have shown improved reconstruction quality for imaging inverse problems [97, 187, 148]; however, they rely on having a large dataset of experimental labeled ground truth and measurement pairs. In these methods, a deep neural network is used to approximate the imaging inverse problem, Fig. 6.1(b). The network is typically trained end-to-end and requires large datasets of labeled image and ground truth pairs. The network can be agnostic to the imaging system physics, or the network can incorporate knowledge of the physics for faster convergence and reduced training data requirements [156, 105]. For high-dimensional imaging applications, such as 3D and hyperspectral lensless imaging, obtaining ground truth datasets can be impractical, costly, or impossible. While there has been some work in training networks from synthetic data, these methods can suffer from model-mismatch if the synthetic data does not match the experimental system [232].

Recent work using unsupervised learning with untrained networks is especially promising for a number of imaging applications—leveraging the structure of neural networks without needing any training data. Untrained networks, such as deep image prior [204] and deep decoder [88], have shown that the structure of neural networks can be effective at serving as a prior on image statistics without any training. An untrained deep network with randomly initialized weights is used as an image generator, outputting the recovered image. The network weights are updated through a loss function comparing the generated image with the input data, for example, a noisy image. This method and several related papers have been shown to be particularly effective for simulated image denoising, deblurring, and super-resolution [147, 133]. For many computational imaging problems, the measurements do not resemble the reconstructed image; instead, the scene is related to the measurement through a forward model that describes the physics of the image formation problem. For this class of problems, untrained networks can be paired with differentiable imaging forward models in which the network weights are updated based on a loss between the measurement and

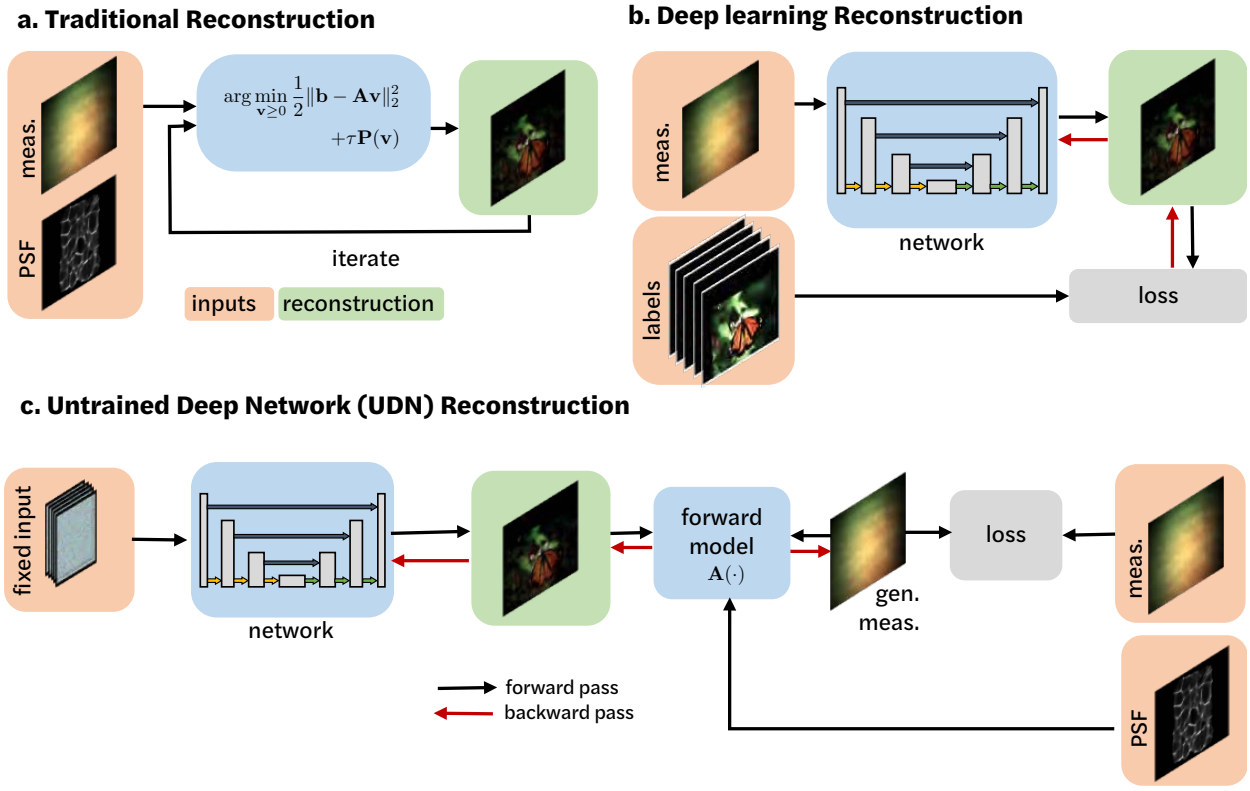


Figure 6.1: **Comparison of different reconstruction approaches**. (a) Traditional reconstructions solve a convex optimization problem with a data-fidelity-term based on the known forward model and a regularization term which acts as the image prior. (b) Deep learning methods require thousands or more labeled image pairs in order to train a neural network to approximate the reconstruction. (c) Our untrained deep network (UDN) uses a neural network as the image prior, but does not require any training data. The system forward model is used to calculate the loss between the estimated image and the measurement, which is then used to update the network weights.

the generated measurement from the network output passed through the imaging forward model, Fig. 6.1(c). This has been shown to be effective on several imaging modalities, such as phase imaging [25], MRI [76, 49, 98], diffraction tomography [236], and several small-scale compressive sensing problems [206]. Here, we extend this framework to compressive lensless photography.

We propose a reconstruction method for compressive lensless imaging based on untrained networks, which we call untrained deep network (UDN) reconstructions. Our approach uses a deep network for the image prior, but requires no training data. We present a general differentiable imaging model that could be used for multiple types of compressive lensless imaging, and test it on three different cases: 2D lensless imaging with erasures, single-shot video, and single-shot hyperspectral imaging.

To the best of our knowledge, untrained networks have not been used for compressive optical imaging before, thus providing both a stress test of untrained deep networks on several challenging underdetermined experimental systems, as well as bringing improved image quality to compressive lensless imaging. We provide simulation and experimental results, showing improved performance over existing methods that do not utilize training data. Our results indicate that untrained networks can serve as effective image priors for these systems, providing better reconstructions in cases where it is not possible to obtain labeled ground truth data.

## 6.1 Imaging Inverse Problem

We demonstrate our UDN on two examples of three-dimensional data recovery from a single-shot 2D measurement: (1) high-speed video [9] and (2) hyperspectral imaging [158]. The high-speed video example takes advantage of the sensor's built-in rolling shutter, which exposes different rows of pixels at each time point, Fig. 3.3(b), effectively acting like an erasure pattern at each time point. From this data, one can use compressed sensing to recover a separate 2D image for each time point; the result is a full-resolution video at the framerate of the rolling shutter (the line scan rate). For the hyperspectral imaging example, the DiffuserCam device has an added spectral filter array in front of the sensor, with 64 different color channels, Fig. 3.3(b). Each of the 64 wavelengths thus map to a different subset of the sensor pixels, similarly acting like an erasure pattern for each wavelength. From this data, one can use compressed sensing to recover a hyperspectral datacube (2D spatial + 1D spectral) from a single 2D measurement. In both examples, we choose to reconstruct the full datacube simultaneously, rather than a sequential set of 2D reconstructions, since this allows us to add priors along the time/wavelength dimension. For more information on the forward model for these cameras, see Ch. 3. The forward models for these cameras will be used to create the differential forward model within our network architecture.

Given our sensor measurement $\mathbf{b}$, our goal is to recover the scene $\mathbf{v}$. For the 2D erasures scenario $\mathbf{v}$ is a 2D image, whereas for single-shot video $\mathbf{v}$ is a video consisting of two spatial and one temporal dimension, and for single-shot hyperspectral $\mathbf{v}$ is a hyperspectral volume consisting of two spatial and one spectral dimension. In all cases, the problem is underdetermined, since we aim to recover more than we measure. First, we describe the traditional reconstruction methods based on convex optimization which will serve as our baseline comparison, and then we describe our untrained network reconstruction method.

### Traditional inverse problem

Due to incoherence of the measurement system that comes from our use of a multiplexing phase optic (diffuser), compressive sensing can be utilized to recover the image. By compressive sensing theory, we can formulate our inverse problem as follows:

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}\geq 0} \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{v}\|_2^2 + \tau R(\mathbf{v}), \tag{6.1}$$

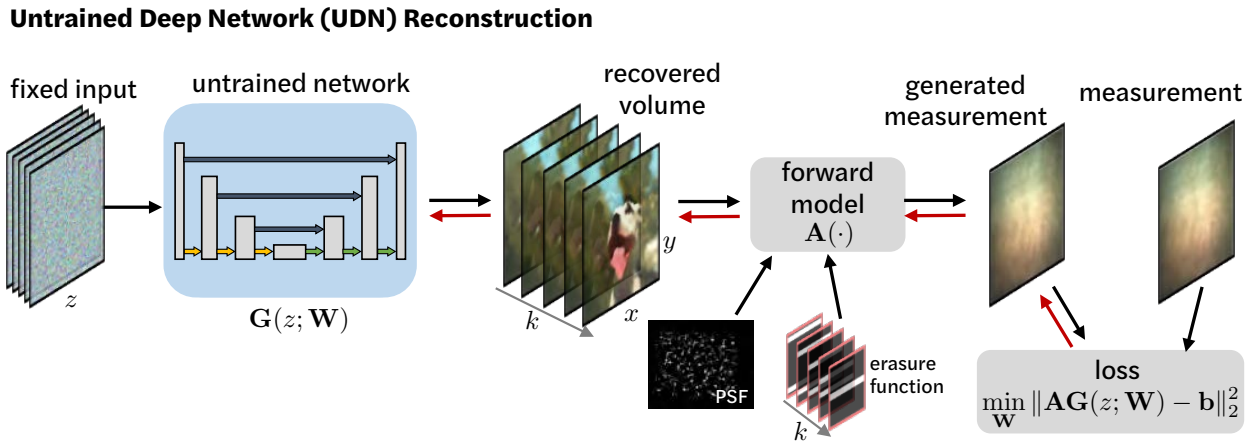


Figure 6.2: **Overview of our Untrained Deep Network (UDN) reconstruction pipeline**. An untrained network with randomly initialized weights takes in a fixed, random input vector. The network outputs a sequence of images along the $k$-axis (for video imaging, one image for each time point), and we pass this output volume into our known imaging forward model (defined by the PSF calibration image and known erasure function) to generate a single simulated measurement. We compare the generated measurement with our captured measurement and use the difference between the two to update the untrained network parameters.

where $R(\mathbf{v})$ is a prior on the scene, and is often of the form $\|\mathbf{Dv}\|_1$, where $\mathbf{D}$ is a sparsifying transform. $\mathbf{A}$ is our forward model, which can be of the form $\mathbf{A}_{\text{2D}}$, $\mathbf{A}_{\text{2D erasures}}$, or $\mathbf{A}_{\text{k-D}}$.

In practice, 2D or 3D TV priors work well for a variety of scenes, and are implemented by defining the regularizer term as $R(\mathbf{v}) = \|\nabla_{xyk}\mathbf{v}\|_1$, where $\nabla_{xyk} = [\nabla_x \nabla_y \nabla_k]^T$ is the matrix of forward finite differences in the $x$, $y$, and $k$ directions. Convex optimization approaches such as fast iterative shrinkage-thresholding algorithm (FISTA) [20] or alternating direction method of multipliers (ADMM) [28] can be used to efficiently solve this problem. In each case, the prior is hand-tuned for the given application by varying the amount of regularization through the tuning parameter $\tau$, which trades off data-fidelity and regularization. As a baseline comparison, we use FISTA with 2DTV for our 2D imaging with erasures problem and weighted anisotropic 3DTV for our single-shot video and single-shot hyperspectral imaging [100]. We include an additional comparison using PnP-BM3D, PnP-BM4D [143], and a pretrained PnP denoiser network [230] in Appendix A.2.

## Untrained network

In this work, we propose to instead use an untrained network for the image reconstruction. It has been shown that neural networks are good at representing and generating natural images; thus, minimizing an image generator network instead of directly solving for the image could be a good way to effectively regularize with a custom deep prior.

Rather than solving for the image $\mathbf{v}$ directly as before, we solve for the image indirectly as the output of the generator network $G(z;\mathbf{W})$. This output image $\mathbf{v}_{gen}$ is then passed through our imaging forward model $\mathbf{A}$ and compared against the measured image $\mathbf{b}$. The network weights are updated based on the difference between the generated measurement and the true measurement, which corresponds to solving the following problem:

$$\mathbf{W}* = \arg\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{G}(z;\mathbf{W})\|_2^2 \tag{6.2}$$

$$= \arg\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{v}_{gen}\|_2^2, \tag{6.3}$$

where our network $G(z;\mathbf{W})$ has a fixed input $z$ and randomly initialized weights $\mathbf{W}$. The network output $\mathbf{v}_{gen}$ is the reconstructed image. $z$ can be thought of as a latent code to our generator network and is randomly initialized and held constant. We update the weights of the network via backpropogation in order to minimize the loss between the actual measurement and the generated measurement (Fig. 6.2). This process must be repeated for each image reconstruction, since there are no 'training' and 'testing' phases as there are for deep learning methods with labeled data.

As in [204], we utilize an encoder-decoder architecture with skip connections and keep the input to the network fixed. See Appendix A.2 for details on our network architecture and hyperparameters for each experiment.

## 6.2 Implementation Details

For the 2D erasures case, we utilize an existing 2D lensless imaging dataset which consists of pairs of 2D lensless camera measurements and corresponding lensed camera ground truth images [156]. This dataset is advantageous because it includes experimental lensless measurements with real sensor noise and other non-idealities, but also provides labeled ground truth data from an aligned lensed camera. For our simulation results, we utilize the ground truth images from the dataset to generate simulated measurements using the forward model in Eq. 3.3; for our experimental results, we directly use the lensless camera measurements. For both, we synthetically add erasures to the measurement by point-wise multiplying it with an erasure function with 0%, 50%, 90%, 95%, and 99% erasures, picking a random subset of indices to erase.

For single-shot video and single-shot hyperspectral imaging, there does not exist experimental data with associated ground truth. For our experimental data analysis, we utilize experimental raw sensor measurements from [9] and [158]. This includes measurements taken from a PCO Edge 5.5 sCMOS camera with a dual rolling shutter and a homemade random diffuser (for single-shot video), and measurements from a hyperspectral DiffuserCam with a spectral filter array and Luminit diffuser [140] (for hyperspectral imaging). For our simulations, we use the PSF and shutter/filter functions from these systems to simulate our sensor measurements.

We implement our network and differentiable forward model with mean square error (MSE) loss in Pytorch and run our reconstructions on a Titan X GPU with the ADAM optimizer throughout training. We perform early stopping to obtain the best reconstructions, as described in [204], with reconstructions ranging from 1,000–100,000 iterations, which generally takes several hours. This process must be completed for every new image; unlike with deep learning methods that use training data, there is no distinction between training and testing and instead the network parameters must be re-optimized for every reconstruction. For reproducibility, training code is available here: *https://github.com/Waller-Lab/UDN/* .

## 6.3 Results

We compare the results of our untrained reconstruction method against the standard FISTA reconstruction with TV regularization. For all three cases, we present both simulation results as well as experimental validation. For the FISTA reconstructions, we reconstructed images with three different amounts of TV regularization, since more regularization leads to improved image quality and lower mean squared error, but tends to blur high-frequency information. Less regularization reveals high-frequency information, but at the price of increased reconstruction artifacts. When ground truth is available (in all simulations and in experiment for 2D erasures), we compare our reconstructed images to the ground truth via a variety of quality metrics: mean squared error (MSE), learned perceptual similarity metric (LPIPS) based on AlexNet and VGG [231], the structural similarity index measure (SSIM), and the multi-scale structural similarity measure (MS-SSIM) [214]. MSE is a standard image metric, but tends to favor low-frequency information. SSIM and MS-SSIM are both perception-based metrics, with MS-SSIM using multiple image scales and achieving better perceptual performance than SSIM. Both LPIPS metrics are learned perpetual similarity metrics based on deep networks, with LPIPS VGG being closer to a traditional perceptual similarly metric. Each method has its strengths and weaknesses, and therefore we provide comparisons using each of the metrics when possible (MS-SSIM requires a certain minimum image size which precludes us from using it for the single-shot video and single-shot hyperspectral cases, and LPIPS only works on RGB color images, so we cannot use it for single-shot hyperspectral). For MSE, LPIPS Alex, and LPIPS VGG, a lower score is better, whereas for SSIM and MS-SSIM, a higher score is better.

### 2D compressive imaging

#### Simulation

First, we simulate a noise-free 2D measurement with increasing numbers of randomly-distributed pixel erasures (0%, 50%, 90%, 95%, and 99%), then compare reconstruction results using both FISTA and our UDN (Fig. 6.3). FISTA uses high, medium, and low amounts of TV corresponding to $\tau = 10^{-4}$, $5.5 \times 10^{-4}$, $10^{-5}$, respectively.

We compare our performance against the ground truth images on our five image quality metrics for an increasing percentage of erasures in Fig. 6.3(b) and show the corresponding

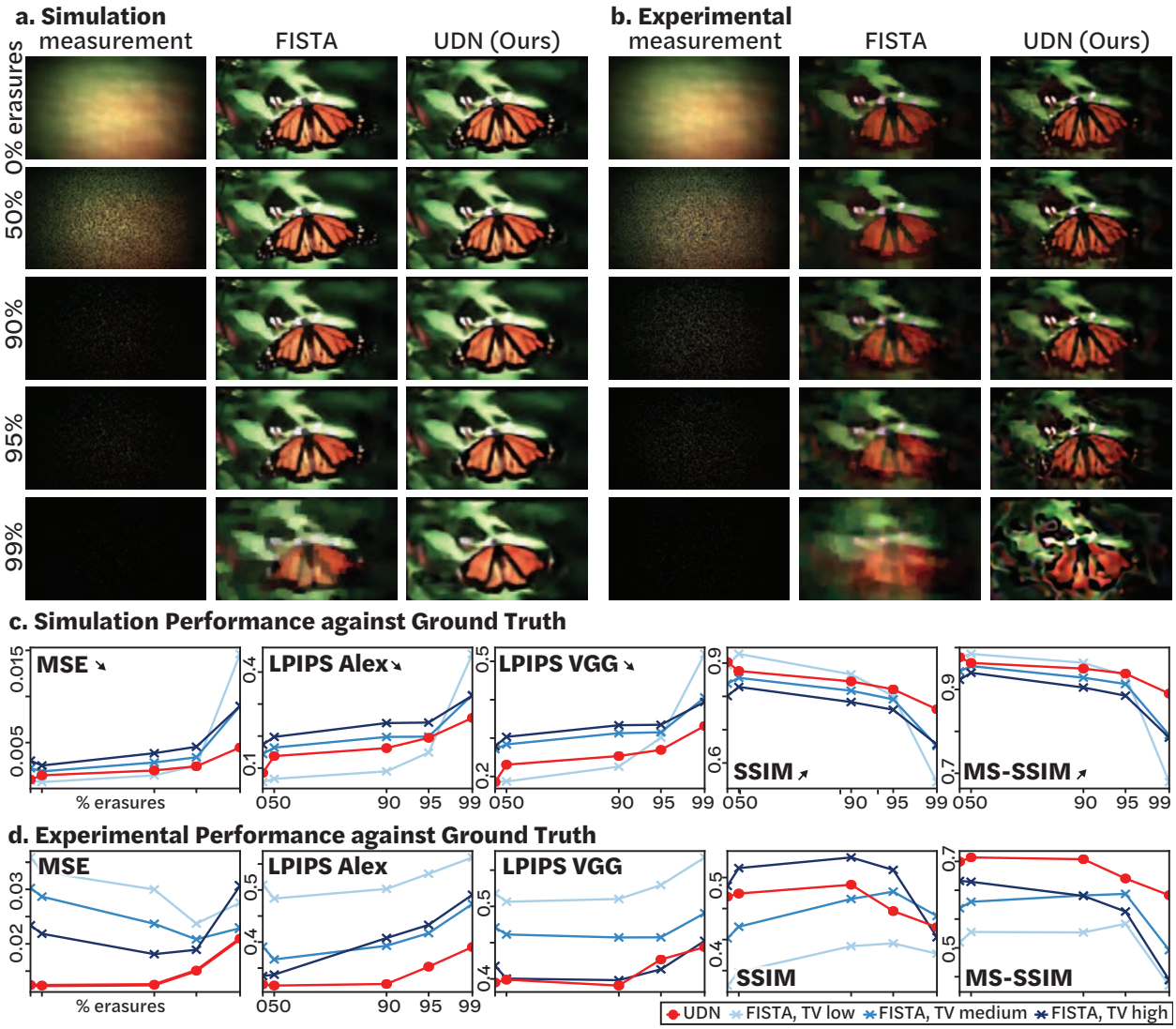


Figure 6.3: **2D Compressive imaging with erasures: simulation and experimental results**. Reconstruction results for increasing numbers of pixel erasures, showing the measurement along with the best fast iterative shrinkage-thresholding algorithm (FISTA) reconstruction (based on the LPIPS Alex metric) and the UDN reconstruction for (a) simulated measurements and (b) experimental measurements with synthetically added erasures. (c) and (d) compare our performance against the ground truth for several quality metrics (arrows indicate which direction is better), showing that in simulation UDN outperforms FISTA only at 99% erasures, whereas in experiments with real noise and non-idealities, UDN outperforms FISTA on the perceptual image metrics, LPIPS Alex and MS-SSIM, as well as on MSE.

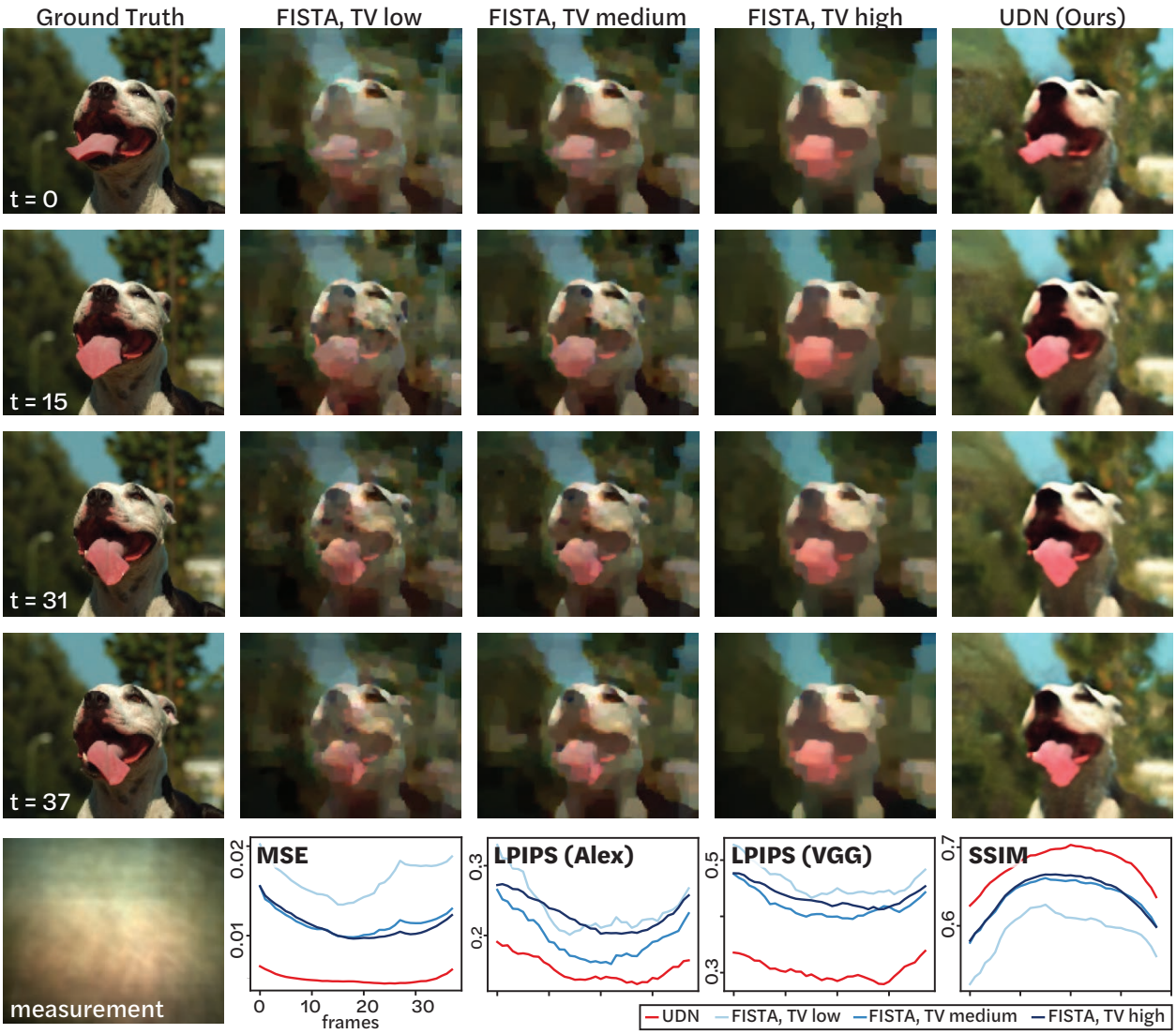


Figure 6.4: **Simulation results on single-shot video**. We recover a 38-frame video from a single measurement (bottom left). We show four sample reconstructed video frames in the top four rows, and plot the quality metrics for all frames below. Here we see that our reconstruction has sharper features across frames, enabling superior recovery especially for the first and last frames, where FISTA has more pronounced reconstruction artifacts. We achieve better MSE, LPIPS, and SSIM scores across frames (bottom right). See Visualization 1 for the full video.

images in Fig. 6.3(a) for our method vs. the best FISTA result (based on the LPIPS-Alex metric). From this, we can see that FISTA and our method perform similarly well for 0%-95% erasures, but our method has improved performance for 99% erasures. This shows that TV regularization is a sufficient prior for smaller numbers of erasures in the absence of noise or model non-idealities, but as the problem becomes severely underdetermined, our UDN provides a better image prior than TV and leads to improved image quality on all of our performance metrics.

### Experimental

Using experimentally captured images from [156], we synthetically add random pixel erasures to the measured data and compare our reconstructions against the ground truth lensed images using the five image metrics described above. As a baseline, we compare against FISTA with high, medium, and low amounts of TV ($\tau$= 0.1, 0.01, 0.001), shown in Fig. 6.3(d). A visual comparison with the best FISTA reconstruction (based on the LPIPS-Alex metric) is shown in Fig. 6.3(c). Unlike in the noise-free simulation, our method consistently performs better than FISTA for all amounts of erasures on the MSE, LPIPS Alex, and MS-SSIM metric, resulting in a clearer and sharper reconstruction. Thus, our method gives better performance improvements over FISTA for real-world datasets, likely because the UDN provides a better image prior and outperforms TV in the presence of real measurement noise and imaging non-idealities.

TV is a very commonly used prior that is computationally efficient, but other priors can be incorporated using the PnP framework. See Appendix A.2 for additional comparisons against PnP-BM3D and a PnP pretrained denoiser network, demonstrating that the UDN outperforms these methods as well.

## Single-shot Video

### Simulation

Using an experimentally captured PSF and shutter function from [9], we simulate a compressive video measurement, shown in Fig. 6.4 top left, using a 38-frame video. Fig. 6.4 shows the results of our reconstruction compared to the FISTA result for several frames of the video (with $\tau = 10^{-2}$, $10^{-3}$, $5 \times 10^{-4}$ for FISTA). It is evident that our method generates a more visually appealing result with fewer artifacts, while preserving high-resolution features throughout the frames. This is quantified in Fig. 6.4(bottom), showing that UDN has a significantly better performance on all metrics. For each of the FISTA results, the values are worse at the beginning and end of the video sequence, which is consistent with [9]. Our method similarly has worse performance for the first and last frames, but the difference is not as pronounced, resulting in more uniform image quality throughout the recovered video.

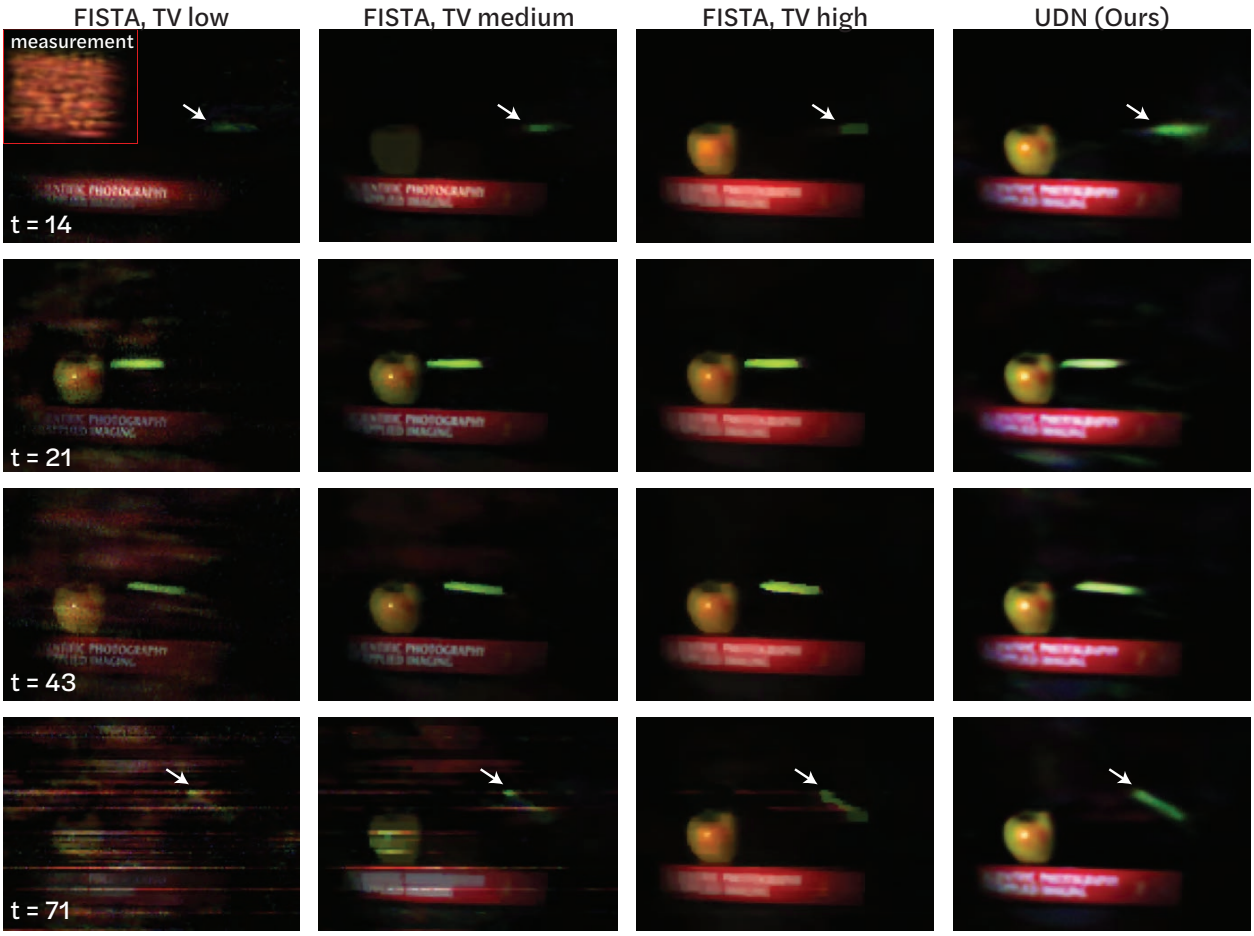


Figure 6.5: **Experimental results on single-shot video**. We recover 72 frames from a single measurement (top left). The rows show four sample reconstructed frames from our 72-frame reconstruction, with both our UDN method and FISTA with three different amounts of TV. Here we see that our reconstruction has sharper features across frames and better captures motion within the video. See Visualization 2 for the full video and Visualization 3 for a second experimental example.

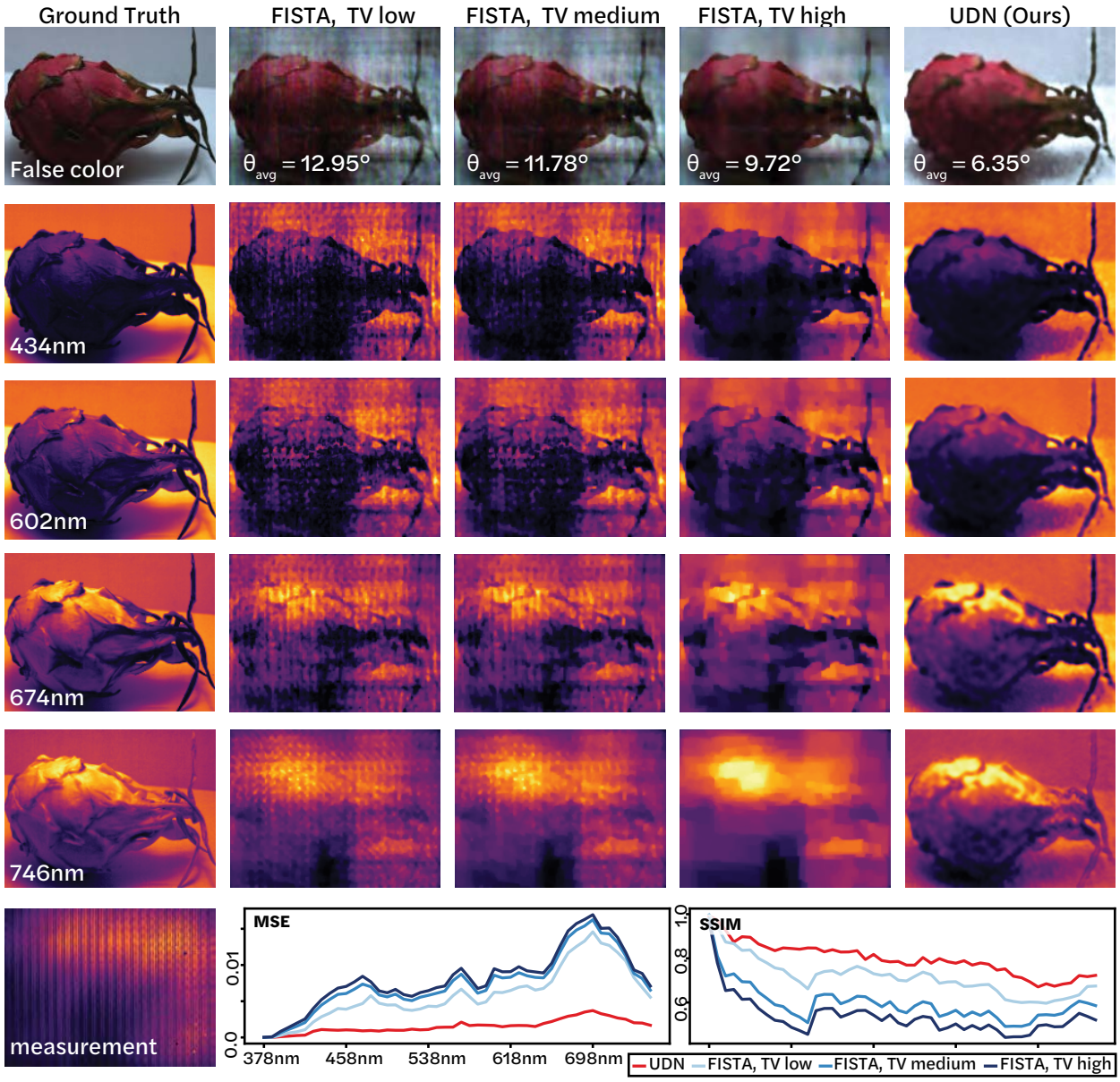


Figure 6.6: **Simulation results on hyperspectral data**. We display a false-color image of the recovered 64-channel hyperspectral volume (top row), along with four selected spectral slices. The quality metrics are plotted for each wavelength at bottom right. Here we see that our reconstruction has sharper features and fewer artifacts than FISTA, and achieves a better MSE and SSIM score across all wavelengths. In addition, UDN achieves better cosine similarity ($\theta_{\text{avg}}$) to the ground truth spectra than FISTA. See Visualization 4 for the full hyperspectral volume.

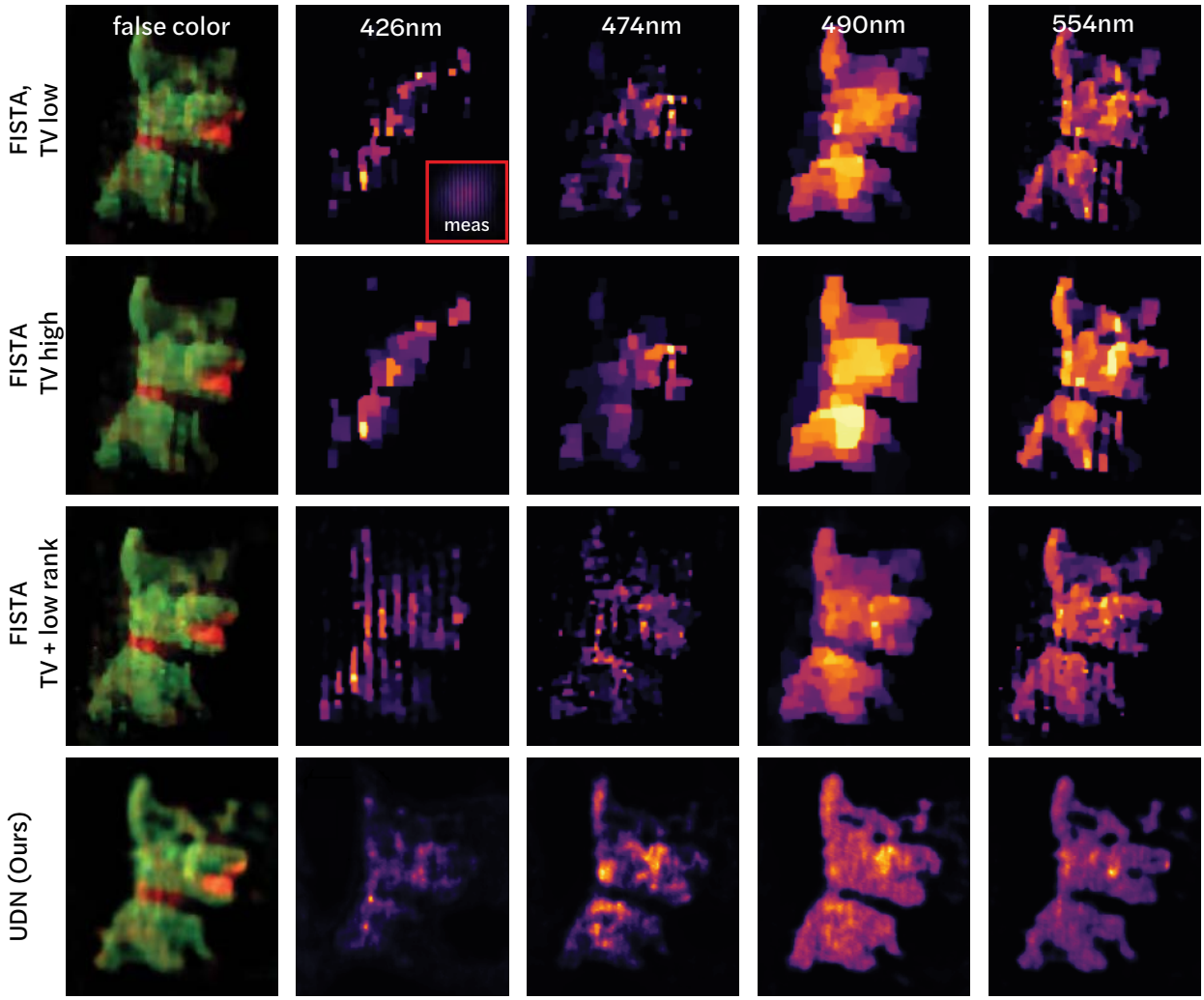


Figure 6.7: **Experimental results on hyperspectral data**. We show a false-color image of the recovered 32-channel hyperspectral volume (top row), along with four spectral slices. Here we see that our reconstruction has sharper features and fewer artifacts across wavelengths. See Visualization 5 for the full hyperspectral volume.

### Experimental

Next, we recover 72-frame videos from a single experimental measurement, Fig. 6.5. When compared to the FISTA reconstructions (with $\tau = 0.01, 0.001, 0.0001$), we can see that our method has more uniform image quality throughout. The foam dart has significant artifacts in the FISTA reconstruction for the first and last scenes of the video, even disappearing for low TV. Our method appears to capture the dart motion throughout the video frames and has fewer noticeable artifacts than the FISTA reconstruction. There is no ground truth available for this data, but our method seems to produce a more realistic and visually

appealing reconstruction than FISTA in terms of the dart motion. FISTA with smaller amounts of TV is able to better recover the text on the book in the scene, but at the expense of reduced image quality and more artifacts throughout the video. Our method is not quite able to recover the book text, but has more uniform image quality and better captures the motion in the scene.

### Single-shot hyperspectral

#### Simulation

Finally, we recover a hyperspectral volume with 64 wavelengths from a single simulated measurement. We simulate the measurement using an experimentally captured PSF and filter function from [158], along with a ground truth hyperspectral image from [63] with 64 spectral channels. Figure 6.6 shows the results of our reconstruction compared to FISTA with high, medium, and low amounts of TV ($\tau = 3\times10^{-7}$, $6\times10^{-7}$, and $3\times10^{-6}$). We can see that UDN preserves more features across wavelengths and has fewer artifacts than FISTA. We compare the MSE and SSIM across the methods, Fig. 6.6(bottom). We can see that UDN has significantly better MSE and SSIM values than FISTA across wavelengths. In addition, we report the average cosine distance between the spectral profiles in the reconstruction and ground truth images (Fig. 6.6). The cosine distance is especially important for hyperspectral imaging due to its role in hyperspectral classification. UDN provides a lower average cosine distance than FISTA, indicating that UDN achieves a better recovery of the spectral profiles. We note that FISTA with high TV achieves a better cosine distance than FISTA with low TV, but at the price of worse spatial resolution. Our UDN method achieves both better spatial quality (MSE and SSIM) as well as better spectral performance (cosine distance) than FISTA.

#### Experimental

We test our performance on an experimentally captured measurement [158] of a Thorlabs plush dog illuminated by a broadband lightsource, recovering 32 spectral channels from a single measurement (downsampled 2× spectrally). We compare against FISTA with low and high TV ($\tau = 3\times10^{-4}$ and $5\times10^{-5}$) along with the best reconstruction from [158] (FISTA TV + low rank), down-sized to match our reconstruction size. While ground truth images do not exist for this data, UDN appears to provide more consistent image quality and retains more of the details across wavelengths than FISTA, Fig. 6.7.

## 6.4 Discussion

Untrained networks offer a number of distinct advantages for compressive lensless imaging. First, they do not require any training data, as opposed to deep learning-based methods. Training data is especially hard or impossible to acquire for higher-dimensional imaging, such as for high-speed video, hyperspectral, or 3D imaging, so this feature is particularly useful.

Untrained networks can serve as a better prior for certain high-dimensional imaging problems off-the-shelf, potentially enabling better reconstructions for a number of compressive imaging modalities.

Currently, the main limitations of untrained networks are: 1) memory-constraints and 2) speed. First, many of our reconstructions are GPU memory limited. To take advantage of accelerated GPU processing, the entire untrained network must fit in memory on the GPU, limiting the size of reconstructions that we can process, and we find that we can process much larger images and volumes with FISTA than we can with UDN. In this work, our single-shot video and single-shot hyperspectral measurements are downsized between 2–16$\times$ from the original size in order to fit in the GPU, limiting our resolution. Looking forward, larger GPU sizes or clever computational techniques to better utilize GPUs for memory-limited problems could improve this and enable the reconstruction of larger images and volumes. Next, the speed of the untrained network reconstructions is generally an order-of-magnitude slower than standard FISTA reconstructions. Thus, our method is best suited for applications where a real-time reconstruction is not needed, since typical untrained reconstructions take between 1–5 hours. As machine learning speeds and processors improve, we expect these factors to be less limiting.

Given the benefits and limitations of untrained networks, we envision this reconstruction method to be useful for a certain subset of problem where it is difficult or impossible to obtain ground truth data for training, but where there are little or no time constraints on the reconstruction. Given the fact that no training data is needed, this method can be applied to many different imaging problems without needing to collect new datasets. UDNs are especially promising for imaging dense, natural scenes where TV can be a poor prior.

One interesting open problem for untrained reconstructions is the network choice. In our experiments, we utilized a convolutional network with an encoder-decoder structure and found that this worked well; however, our network tended to blur out high frequency features and there may be other architectures that could potentially provide better priors. For instance, deep decoder [88] has been demonstrated for similar tasks, but we found that for our application it required more iterations to converge and was outperformed by an encoder-decoder structure, demonstrating that the choice of network architecture is important and application-specific. Although our network worked well for photographic scenes, other network architectures may be more appropriate for other types of images, such as fluorescent biological targets which may have very different statistics than photographic scenes.

## 6.5 Conclusion

We have demonstrated that untrained networks can improve the image quality for lensless compressive imaging systems. We tested untrained networks on 2D lensless imaging with varying amounts of erasures, and we demonstrated their effectiveness on single-shot compressive video and single-shot hyperspectral imaging, in which we recover a full 72-frame video or 32 to 64 spectral slices, respectively, from a single measurement. In each case, we showed both in simulation and experiment that untrained networks can have better image quality

than compressive-sensing-based minimization using total-variation regularization, demonstrating that non-linear networks can be a better prior than TV for dense, natural scenes. We believe that untrained networks are especially promising for situations in which training data is difficult or impossible to obtain, providing a better imaging prior for underdetermined reconstructions.

# Chapter 7

# Learning a Physics-informed Noise Model for Extreme Low-light Imaging

In this chapter[1], we formulate a physics-informed, low light noise model for extreme low light videography. Noise in extremely low light, high gain sensor settings is non-Gaussian and hard to quantify. To tackle this, we formulate a noise generator that, after training, can generate realistic sensor noise. This noise generator is based on a generative adversarial network, but is constrained to learn a few physics-informed noise parameters that are commonly used to quantify sensor noise. We then apply this noise generator to synthesize a dataset of noiseless/noisy videos with significant motion, and use this to train a video denoiser that operates at submillilux (starlight) levels of illumination, showing photorealistic videos in starlight for the first time.

Some animals, such as hawkmoths and carpenter bees, can effectively navigate on the darkest moonless nights by the light of the stars ($< 0.001$ lx) [190, 217, 101], while our best CMOS cameras generally require at least 3/4 moon illumination ($> 0.1$ lx) to image moving objects at night [34]. Seeing in the darkest settings (moonless, clear nights) is extremely challenging due to the minuscule amounts of light present in the environment. In such dark settings, photographers can use long exposure times (e.g. 20s or higher) to collect enough light from the scene. This approach works well for still images, but severely limits the temporal resolution and precludes imaging of moving objects. Alternatively, cameras can increase the gain, making each pixel effectively more sensitive to light. This allows shorter exposures, but greatly increases the noise present in each frame. In this setting, motion might be perceptible, but noise overwhelms the images.

[1]This chapter is based on the published conference paper titled "Dancing under the stars: video denoising in starlight" and is joint work with Stephan Richter, Laura Waller, and Vladlen Koltun [155].

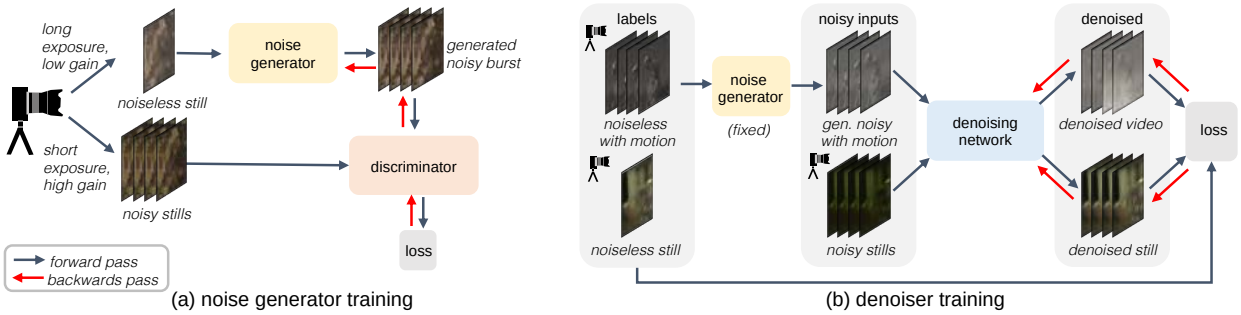


Figure 7.2: **Method overview**. (a) First we train our noise generator along with a discriminator, which aims to distinguish between real and synthetic noise. We use a limited dataset of long exposure/low gain and short exposure/high gain non-moving image pairs during this training process. After training, the noise generator can synthesize realistic noise. (b) Next, we train our denoiser using a combination of synthetic clean/noisy video clips produced using our noise generator as well as real still clips from our camera. This allows us to train a video denoiser without needing experimental motion-aligned video pairs.


Figure 7.1: **Video denoising in submillilux**. (a) Raw noisy video clip (10fps video) taken between 0.6–0.7 mlx (millilux) on a clear, moonless night with no external illumination. (b) Result after contrast stretching the video clip. (c) Denoised video using our denoiser.

Denoising algorithms can be used to improve the image quality in noisy images. Over the years, a number of denoising algorithms have been developed, from classic methods (e.g. BM3D [50], V-BM4D [144]) to deep learning based approaches [229]. Each of these methods attempt to extract the signal from the noise based on some assumptions about the distribution of the image and noise. While successful for certain denoising tasks, most of these methods are built upon a simplistic noise model (Gaussian or Poisson-Gaussian noise), which does not apply in extremely low light settings. When high sensor gain is used in low light images, the noise is often non-Gaussian, non-linear, sensor-specific, and difficult to model or characterize. Without having a good understanding of the structure of the noise in the images, denoising algorithms may fail to denoise the image – mistaking the structured noise for a real signal in the image.

Recently, several deep learning-based approaches have provided remarkable denoising performance in low light down to 0.1–0.3 lux [44, 45]. Rather than assuming a certain noise model, these methods trained a denoiser using real clean/noisy image pairs from a camera. Such an approach automatically accounts for the low light noise through deep learning, however this comes at the price of needing to capture thousands of training image pairs. Such a dataset is camera-dependent and would need to be retaken for a different sensor, since noise can be highly camera-specific. In addition, while it is possible to capture clean/noisy image pairs for non-moving objects by changing the exposure/gain settings, capturing clean/noisy image pairs for moving scenes adds additional complexities (e.g. needing a second camera, aligning motion), making this experimentally impractical [96].

To achieve submillilux video denoising for the first time, we propose to use a combination of three things: 1) a very good camera optimized for low-light imaging and set to the highest gain setting (Sec. 7.3), 2) learning our camera's noise model using a physics-inspired noise generator and easy to obtain still noisy images from our camera (Sec. 7.2), and 3) using this noise model to generate synthetic clean/noisy video pairs to train a video denoiser (Sec. 7.4). Since our physics-based noise generator is trained using a limited dataset of still clean/noisy burst, we do not need to acquire experimental motion-aligned clean/noisy video clips, greatly simplifying the experimental setup and decreasing the amount of data we need to collect. After noise generator training, we hold the noise generator fixed and train our video denoising using a combination of real still clean/noisy image bursts paired with synthetic video clips (Sec. 7.4). Fig. 7.2 summarizes this two-stage training approach for our noise generator and denoiser.

We demonstrate the effectiveness of our denoising network on 5–10fps videos taken on a moonless clear night in 0.6 mlx, showing photorealistic video denoising in submillilux levels of illumination for the first time. We present several challenging scenes with extensive motion, in which subjects dance by only the light of the milky way as a meteor shower rains down from above.

## 7.1 Related Work

**Image and video denoising.** A variety of techniques for image and video denoising have been proposed and studied throughout the years. Many of the classic denoising methods rely on specific image priors, such as sparsity [170, 61], smoothness [177], or Gaussian mixture models [224, 57]. Others utilize a non-local strategy to collaboratively denoise similar patches across an image [31, 118, 50, 144]. More recently, deep learning based approaches have been applied to image denoising, in which an image prior is learned from the data rather than explicitly assumed [33, 60, 48, 200, 205]. These methods have shown significant improvements over classic methods in terms of image quality, however they often make simplistic assumptions on the noise statistics, such as i.i.d Gaussian. When trained with these simplistic assumptions, classic techniques such as BM3D often outperform deep learning based methods on real photographs with real noise [169]. In this regard, several datasets of noisy and clean image pairs from real cameras have been created to benchmark and improve

the performance of deep learning-based denoisers on real cameras [6, 169, 3]. In addition, some work has focused on "unprocessing" online image datasets to better match RAW image distributions in order to generate more synthetic data for training RAW image denoisers [30].

Alternatively, another line of deep learning based denoising focuses on unsupervised learning, in which no ground truth images are used to train the denoiser. Such methods either assume that the structure of a deep network can act as a prior for the image denoising [204], or assume statistical independence of the noise to train a denoiser using samples drawn from one [19, 110, 111, 171] or multiple [120] noisy image frames. While this line of work is promising since it does not require ground truth data and therefore can be adapted for different camera sensors, these methods are not easily adaptable to the more structured or signal-dependent noise that is present in low light settings under high gain, such as banding noise.

**Low light photography.** A number of methods focus particularly on the challenging case of denoising for low light and night photography. A popular method for low light photography is burst denoising, in which multiple images are merged and denoised. This technique is used in HDR+ as well as Google Night Sight [85, 129]. These methods require robust alignment techniques to account for any motion in the scene, which is difficult in the presence of extreme noise. A number of deep learning-based approaches have emerged that attempt to do this burst-alignment step automatically through deep learning [151, 74]. Often, the final goal of burst denoising is to obtain a single clean image from the noisy burst. In our work, we aim to obtain a full video from a noisy set of images.

Recently, a number of deep learning based methods attempt to address low light photography by learning to denoise images in the presence of extreme noise. These methods learn a denoiser and image enhancer network for underexposed low light images by first collecting a training dataset of clean/noisy image pairs [44, 96, 45]. These methods have demonstrated remarkable results for low light conditions down to 0.1 lx. Our work pushes this limit down by two orders of magnitude, demonstrating video denoising below 1 mlx.

Furthermore, these methods rely on a camera-specific dataset of ground truth/noisy image pairs for training, which is particularly challenging for video denoising. Our approach only requires a limited dataset of still image pairs for video denoising, eliminating the need for a large experimental dataset of noisy/clean aligned videos.

**Noise models.** A Gaussian noise model is commonly used to model noise in an imaging system, however this is not a very realistic representation of real-world sensor noise [169]. Signal-dependent models, such as Poisson-Gaussian [68, 67] or a heteroscedastic Gaussian model [84] are more realistic as they account for the effect of shot noise in cameras. However, there are many more effects, such as clipping [68], fixed pattern noise, and banding noise that these models do not account for [108, 26]. Additional work has focused on characterizing sensor noise in low light environments by fitting to certain distributions for different noise components [212, 218]. In general, noise modeling for extremely low light imaging is complicated and it is difficult to accurately characterize and synthesize realistic camera noise, since the noise can be highly structured and sensor-dependent [16, 108].

Recently, rather than characterizing the noise present in a sensor, several methods have attempted to learn to synthesize realistic noise using generative adversarial networks [46,

202] and normalizing flow models [2]. However, recent work suggests that physics-based statistical methods outperform DNN-based methods [233]. We combine physics-based statistical methods with GAN-based training techniques to learn to approximate the sensor noise in a data-driven manner, without the need for hand-calibrating a noise model.

## 7.2 Physics-inspired Noise Generator

Cameras aim to exactly measure and record the light intensity of a scene, converting photons to voltage readings, which are then converted to bits by an analog to digital converter (ADC). Throughout this process, noise is inadvertently added to the measurement both as a function of photon statistics and the circuitry of the sensor. In well-lit environments, sensor noise is well understood and can be modeled as a combination of two primary components: shot noise, which originates from photon arrival statistics, as well as read noise, which is caused by imperfections in the sensor readout circuitry [84]. In low light settings, this noise approximation breaks down and does not adequately describe the complex noise statistics of the scene. Previous work has shown that noise in low-light settings can be expressed as a combination of photon shot noise, read noise, row noise, and quantization noise, which can be obtained through a rigorous calibration process [218]. Inspired by this work, we propose a physics-inspired noise generator which consists of several learned statistical noise parameters. Rather than calibrating the noise parameters by hand, we automatically learn the optimal parameters using a GAN that is fed several pairs of calibration clean (long exposure, low gain)/noisy (short exposure/high gain) image pairs. Using this framework, our noise generator is trained to synthesize realistic noise in extremely low light and high gain settings, see Figure 7.3.

### Physics-inspired parameters

Our noise generator contains several physics-inspired parameters, mainly consisting of variance terms for random distributions, as well as a CNN to capture any additional effects, known or unknown, that are difficult to specifically model. Both components are jointly optimized during training. First, we parameterize the contributions of read and shot noise. Shot noise is a function of the light intensity hitting the sensor and is often modeled as a Poisson random variable, whereas read noise can be approximated as a zero-mean Gaussian random variable [68, 67]. Together, these are commonly approximated using a single heteroscedastic Gaussian random variable, where the mean is equal to the true signal, $x$, and the variance is parameterized by the read, $\lambda_{read}$, and shot noise, $\lambda_{shot}$:

$$N_s + N_r \sim \mathcal{N}(\mu = x, \sigma^2 = \lambda_{read} + \lambda_{shot} x) \tag{7.1}$$

Low-light imaging often suffers from banding noise, which is a camera-dependent noise that results from the camera circuitry and is particularly prominent at high ISO settings. Banding noise often appears as horizontal or vertical lines in the measurement [108, 26]. We model this as a fixed offset added to each column/row, where the fixed offset is drawn from

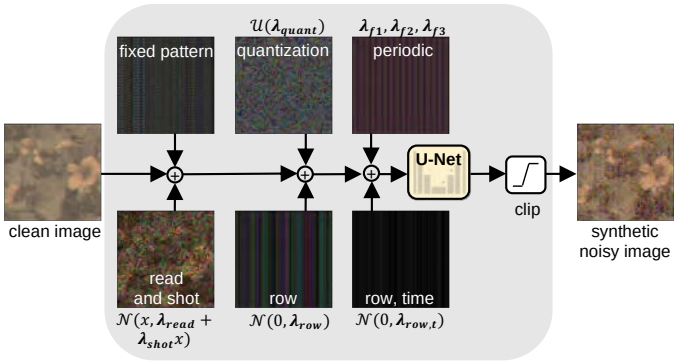


Figure 7.3: **Physics-inspired noise generator**. Our noise generator takes in a clean image and produces a synthetic noisy image. During training, our physics-inspired statistical noise parameters are optimized along with a U-Net to produce a synthetic noisy image that is indistinguishable from a real noisy image.

a zero-mean Gaussian random variable with variance $\lambda_{row}$, see Fig. 7.3. Banding noise is generally independent for each frame, however we have noticed that in extreme low light and high gain settings some banding patterns are consistent across a number of frames. To model this, we also include a time-consistent banding pattern noise which is static across each set of frames. As with the original banding noise, this time-consistent noise is modeled as a zero-mean Gaussian random variable with variance $\lambda_{row,t}$.

In addition to this, at extreme gain settings, we notice that the measurements suffer from periodic noise, potentially due to ADC imperfections/effects at these high gain settings. This periodic noise appears as spikes in the frequency domain of the raw noisy measurements, corresponds to adding a 1 or 2 pixel period sinusoidal pattern to the image with a random amplitude, Fig. 7.3. We parameterize this random amplitude by learned parameters: $\lambda_{f1}$, $\lambda_{f2}$, $\lambda_{f3}$. See Appendix A.3 for further implementation details and discussion.

Next, we add a uniform noise component, to approximate quantization noise in the sensor:

$$N_q \sim \mathcal{U}(\lambda_{quant}) \tag{7.2}$$

Where $\lambda_{quant}$ is our parameter for the quantization noise interval. Generally, this noise component is well-defined based on the number of bits used by the camera sensor. However, we find that allowing this noise parameter to vary can improve our noise generator. Finally, we include a fixed pattern noise component, $N_f$ that stays constant throughout all images. We measure this experimentally using an average of several image sequences. We find that letting this fixed pattern noise be learned can improve the KL divergence between the real noise and generated noise, but this parameter is prone to overfitting and we achieve the best denoising performance when leaving this parameter fixed and experimentally measured.

Thus, our physics-inspired noise model consists of the following components:

$$N = N_s + N_r + N_{row} + N_{row,t} + N_q + N_f + N_p \tag{7.3}$$

Where $N_{shot}, N_{read}, N_{row}, N_{row,t}, N_q, N_f$, and $N_p$ approximate the contributions of shot noise, read noise, row noise, temporal row noise, quantization noise, fixed pattern, and periodic noise.

After initial noise is added to a clean image using the physics-inspired parameters, the intermediate noisy image is passed through a CNN, which aims to improve the initial noise estimate and capture any effects that were not captured by the physics-inspired noise model. We utilize a residual 2D U-Net [176] for this (See Appendix A.3 for architecture details). The final output of our noise generator is clipped to $[0, 1]$. Together, we have a total of 8 physics-inspired parameters ($\lambda_{read}$, $\lambda_{shot}$, $\lambda_{quant}, \lambda_{row}, \lambda_{row_t}, \lambda_{f1}, \lambda_{f2}, \lambda_{f3}$), as well as the parameters of the U-Net. All parameters are optimized during training to produce a realistic synthetic noisy image from a noiseless image. Figure 7.3 shows our physics-guided noise generator with a sample for each noise component.

### GAN training

We want our noise generator to produce different noise samples at each forward pass. This is incompatible with direct supervision where each clean image would be paired to a ground truth noisy image. Thus, to train our noise generator, we resort to an adversarial setup [77], consisting in this case of our noise generator and a discriminator, which evaluates the realism of synthesized noisy images.

Our discriminator operates on noise patches of size 64. For our training objective, we utilize a standard Wasserstain GAN with a gradient penalty framework [82], which is optimized with the following objective function:

$$L = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[D(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x}\sim\mathbb{P}_{\hat{x}}} \|(\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{7.4}$$

where $\mathbb{P}_r$ is the real noisy data distribution, $\mathbb{P}_g$ is the model distribution defined by the generator, $\tilde{x} = G(z)$, $z$ is a noiseless image patch, and $D$ is our discriminator. See Appendix A.3 for full training details.

## 7.3 Camera Selection and Data Collection

To meet our objective of producing photo-realistic videos at submillilux illumination levels, we need to carefully choose an appropriate camera sensor and lens. Generally, larger pixel sizes are preferable for low-light imaging, since each pixel has more photons hitting it than for a smaller pixel. In addition, near infrared (NIR) sensitivity is useful for nighttime imaging because there are more detectable photons in NIR than at RGB wavelengths at night [121, 208].

We choose to use a Canon LI3030SAI Sensor, which is a 2160 x 1280 sensor with 19$\mu$m pixels, 16 channel analog output, and increased quantum efficiency in NIR. This camera has a Bayer pattern consisting of red, green, blue (RGB), and NIR channels (800–950nm). Each RGB channel has an additional transmittance peak overlapping with the NIR channel

to increase light throughput at night. During daylight, the NIR channel can be subtracted from each RGB channel to produce a color image, however at night when NIR is dominant, subtracting out the NIR channel will remove a large portion of the signal. Due to the prevalence of NIR at night, our night videos often have muffled colors. We pair this sensor with a ZEISS Otus 28mm f/1.4 ZF.2 lens, which we choose due to its large aperture and wide field of view.

We capture 3 sets of datasets from our camera: bursts of paired clean (low gain, long exposure)/noisy (high gain, short exposure) static scenes, clean videos of moving objects, and noisy videos of moving objects in submillilux conditions. The paired dataset of static scenes is used to train our noise generator. Both the paired dataset and the clean videos of moving objects are used to train the denoiser. The final dataset is reserved for testing the performance of our denoiser in the most challenging setting. Our datasets will be open-sourced and can be used as a challenge for future denoising algorithms.

**Paired clean/noisy bursts of static scenes.** We collect 10 clips of grayscale and color targets, each with one clean image along with a burst of 100–900 noisy images, resulting in a dataset with 2558 noisy images. We use this dataset exclusively for the noise generator training. In addition to this, we acquire a more complex dataset of 67 clean/noisy image pairs consisting of 16 noisy bursts for each clean image. This second dataset contains indoor and outdoor scenes with various lighting conditions. We use this dataset both for our noise generator and denoiser training.

**Unpaired clean RGB+NIR videos.** With our trained noise generator, we can generate unlimited amounts of clean/noisy pairs from clean videos. Given an absence of RGB+NIR RAW datasets, we collect our own dataset of noiseless video clips (unpaired). We collect 10 video sequences which we break into 166 video clips for training and 10 for testing. The videos are taken at different frame rates of both indoor and outdoor scenes. We capture these images at a low gain setting during the daytime.

To augment our dataset, we utilize 329 video clips from the MOT video challenge [117], which we then unprocess [30] to resemble RAW video clips. While these video clips have significant motion, they have a different distribution of colors than the raw data from our camera. Due to this, we utilize the MOT videos during our initial pre-training step, then refine our denoiser using only the video clips from our camera.

**Submillilux RGB+NIR videos.** To test our method in the lowest light settings, we collected videos in a remote location on a clear, moonless night (outside of most of the night-glow from cities). Throughout our experiments, no outside light sources were used to illuminate the scene. The illuminace, measured by a PR-810 Prichard Photometer, ranged between 0.6 mlx–0.7 mlx, which is within the range expected for a clear moon-less night. Videos were taken with exposures ranging from 0.1–0.2ms per image, corresponding to 10–5 fps. All videos were taken with the largest lens aperture to maximize the amount of light hitting the sensor, and at the highest gain settings for the camera.

## 7.4 Video Denoising

Now that we can generate clean/video pairs, our next step is to train a denoiser that will generalize well to real noisy video clips from our camera. Inspired by burst-denoising, in which a burst of multiple noisy frames are used together the denoise a central frame, we choose a network architecture that can operate on multiple frames at a time. This is beneficial because denoising a burst of images can improve PSNR over single-image denoising, especially in photon-starved regimes. In addition, video-denoising can help us maintain temporal consistency across frames and reduce flickering throughout the denoised video.

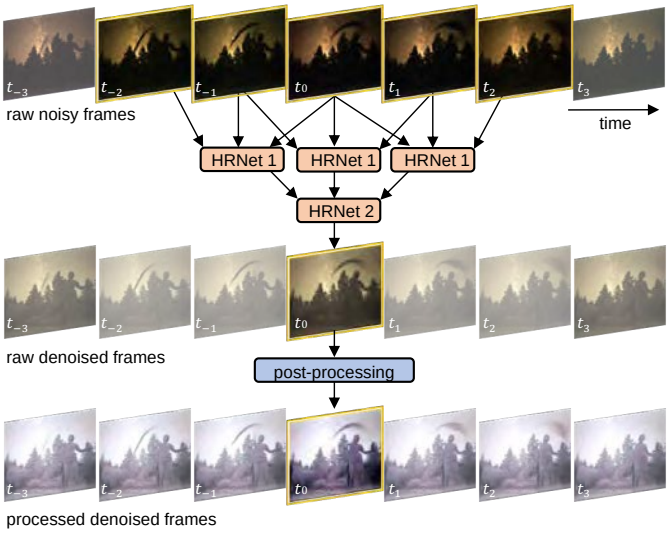


Figure 7.4: **Denoising network**. Our denoising network has a similar overall structure to FastDVDnet, sequentially taking in 5 noisy RAW images to produce 1 denoised RAW image. After denoising, off the shelf post-processing (e.g. white-balancing, histogram equalization) is applied to produce the final denoised video.

### Denoising Network

For our denoising network, we build off of FastDVDNet [200], which is a state of the art video denoiser that implicitly handles motion estimation within the network. We modify this network by replacing the U-Net denoising blocks with an HRNet from [195], which we find leads to better temporal consistency across our denoised video than the original U-Net architecture. Our denoiser operates on RAW video sequences, Fig. 7.4, and off the shelf post-processing is used produce the final output. See Appendix A.3 for our full denoiser network architecture and our evaluation against the original FastDVDNet architecture.

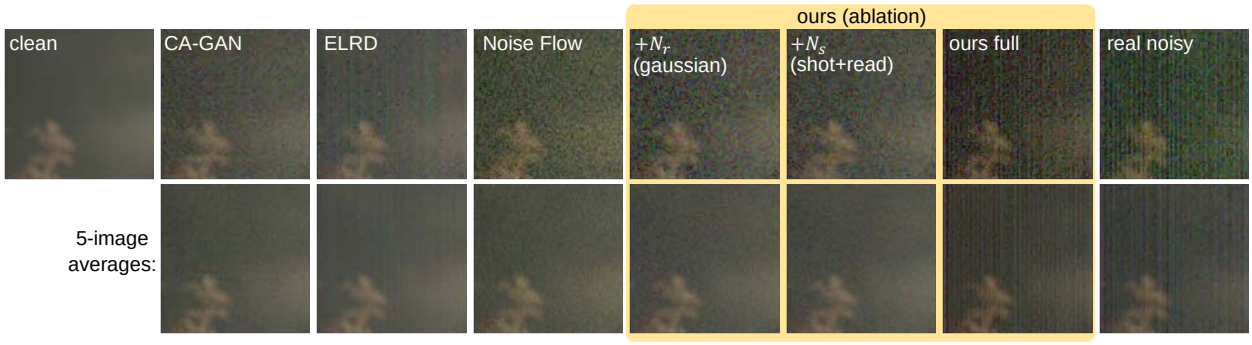


Figure 7.5: **Noise model comparison**. We show an example image patches with our noise model vs. alternative noise models, as well as the mean of this image patch over 5 samples. Our synthetic noise appears more similar to the real noise than alternative methods and closely matches the average noise as well.

## Training

We train the denoiser on a combination of synthetic noisy video clips and real still images from camera. First, we pretrain our network for 500 epochs using a combination of real paired stills, synthetic noisy clips from our camera, and synthetic noisy clips from the MOT dataset to help prevent overfitting. After pretraining, we refine the model for 817 epochs on our real still images and synthetic clips from our camera. All images are cropped to 512×512 patches throughout training. We use a combination a perceptual loss (LPIPS) [231] with an $l_1$ loss for our training objective, choosing only the first 3 RAW channels for the LPIPS loss, which requires a 3-channel image. We gamma correct our ground truth images with gamma= (1/2.2), thereby training the denoising network to output a gamma-corrected image. We found that this outperformed applying the gamma correction after denoising. For both pretraining and refinement, we utilize the Adam optimizer [106] with learning rate $10^{-4}$ and all default parameters.

## Post-processing

Our denoiser is trained on the raw images from the camera. In doing so, this system can work with a number of different post-processing pipelines. To display our final images, we apply the following post-processing steps: demosaicing via bilinear interpolation, white balancing, and histogram equalization. We note that our denoised images are already in a gamma-corrected space. We display the RGB channels of the video clips, omitting the NIR channel in our visualization. We expect that manual post-processing in Adobe Lightroom or a comparable platform could further improve the contrast and perceptual quality of our images.

## 7.5 Evaluation

First, we evaluate our noise generator performance against several existing noise models for low light imaging, as well as perform an ablation on the components of our noise model. Next, we compare our noise generator + video denoiser pipeline against several existing denoising schemes. We quantitatively compare on a held-out dataset of still noisy/clean image pairs. Finally, we qualitatively compare on our submillilux dataset of noisy videos, which do not contain ground truth labels for a quantitative comparison.

### Noise generator

After training our noise generator, we assess its performance on a held-out dataset consisting of 832 128×128 video patches. Each patch has 4 color channels and 5 temporal channels. We calculate the KL divergence between our synthetic noise and the real noisy clips after subtracting the clean image. We compare against a non-deep low light noise model (ELD) [218], as well as two deep-learning-based noise models, CA-GAN [42] and Noise Flow [2]. ELD is hand-calibrated using dark frames and grayscale frames to fit to several distributions for different noise sources. Following this calibration scheme, we found that our noise distributions are very different from those described in [218], likely due to our extremely high gain settings, predominant fixed pattern noise, and periodic components, leading to poor performance of this model. CA-GAN is a camera-aware noise model that takes in a clean image, an estimated shot and read noise image, as well as an example real noisy image from the camera in order to synthesize a signal-dependent noisy image. We use this model off-the-shelf, but find that it generalizes poorly to our camera and noise. Similarly, Noise Flow is designed to work with multiple gain settings and lighting conditions, but does not generalize to our extreme low-light, high gain setting. We summarize our findings in Table 7.1, and show an example synthetic noisy patch from each method in Fig. 7.5. We can clearly see that both Noise Flow and CA-GAN miss the significant banding noise (column offsets) present in our real noisy clips. ELD captures the banding noise pattern well, but misses other components of the noise and does match the real noisy clips visually or in terms of KL divergence.

**Ablation of noise parameters.** We ablate different noise components of our generator in Table 7.1 and show a qualitative comparison in Figure 7.5. As before, we calculate the KL divergence between synthetic and real noisy patches, and we find that each component of our noise model improves the KL divergence. Specifically, shot, read, quantization, and row noise, were all used ELD [218], but were hand-calibrated. Here, we automatically calibrate the different noise components through our GAN training, resulting in better performance than the hand-calibrated model. In addition, our model takes into account the noise behavior over time by including components that are constant over multiple image patches (temporal row noise, fixed pattern noise). As seen in the 5-images averages on the bottom of Figure 7.5, our noise model closely matches the average noise, which is important for video denoising. Adding a periodic noise component makes our noise better match the Fourier spectrum of the real noise, which has several prominent peaks in Fourier space (see Appendix A.3 for details). Adding our measured fixed pattern, $N_f$, improves the behavior over time, and learning the

| Noise model | KLD |
|---|---|
| ELD [218] | 1.361 |
| Noise Flow [2] | 0.386 |
| CA-GAN model [42] | 0.513 |
| Ours (ablation) | |
| $N_r$ (Gaussian) | 0.400 |
| $N_s + N_r$ (shot + read noise) | 0.400 |
| $N_s + N_r + N_q$ | 0.122 |
| $N_s + N_r + N_q + N_{row} + N_{row_t}$ | 0.118 |
| $N_s + N_r + N_q + N_{row} + N_{row,t} + N_p$ | 0.113 |
| $N_s + N_r + N_q + N_{row} + N_{row,t} + N_p + N_f$ | 0.138 |
| $N_s + N_r + N_q + N_{row} + N_{row,t} + N_p + N_{f*}$ | 0.084 |
| Full model: | **0.0691** |

Table 7.1: We compare our noise generator to prior work, representative of different approaches to modeling noise distributions. Our method significantly outperforms all baselines. We also present an ablation of components modeled by our noise generators. See Figure 7.5 for a visual comparison.

fixed pattern, $N_{f*}$ further improves the KL divergence, but at the price of risking over-fitting since this is a pixel-wise addition of an image to the synthetic noise. In our final model, we use a measured fixed pattern and a learned U-Net which can account for noise that we do not specifically model, such as chromatic effects, or enhance our Gaussian noise approximations to better match the true noise distributions. Our final noise model produces synthetic noise that closely matches the real noise for a single noise instance, over time, and in Fourier space.

## Full pipeline: video denoising

Next, we evaluate our video denoiser trained using combination real and synthetic noisy samples against existing denoisers. First, we quantitatively compare against several alternative methods using our dataset of 21 still clean/noisy bursts (since we do not have ground truth for our noisy video clips). We split up our comparison into two categories: single-image denoising methods and video denoising methods, which take in multiple clips at a time.

For single-image denoising, we compare against two pre-trained deep denoising methods: Unprocessing [30], which operates on RAW images and is trained using different read and shot noise levels, as well as L2SID [44] which takes in a raw noisy image and jointly denoises and processes the image. Both of these methods do not perform well on our dataset, due to the extreme noise in our raw measurements. We also retrained L2SID [44] using our still image pairs, resulting in better performance single-images, but significant flickering over time for video (see Appendix A.3 for example). Noise2Self, a self-supervised approach, does

| Method | PSNR | SSIM | LPIPS |
|---|---|---|---|
| **Single Image Methods:** | | | |
| Noise2Self [19] | 20.11 | 0.210 | 0.545 |
| Unprocesing [30] | 12.86 | 0.249 | 0.355 |
| L2SID (pretrained) [44] | 13.6 | 0.512 | 0.338 |
| L2SID (retrained) [44] | 26.9 | 0.892 | 0.198 |
| **Video Methods:** | | | |
| V-BM4D [144] | 16.2 | 0.322 | 0.419 |
| pretrained PaCNet [205] | 13.65 | 0.512 | 0.338 |
| pretrained FastDVDnet [200] | 23.8 | 0.618 | 0.282 |
| **ours** | **27.7** | **0.931** | **0.078** |

Table 7.2: Performance on still images from test set.

poorly with our noise due to its highly structured content (e.g. correlated lines for the row offsets), and results in denoised images with prominent line artifacts. These results are summarized in Table 7.2, with full images shown in Appendix A.3.


Figure 7.6: **Video Results**. Results on noisy video clips taken at 10fps in 0.0006 lx. The input sequence (left), V-BM4D, pretrained FastDVDnet, and our results are shown. Our method maintains more details throughout the clip and does not contain the prevalent streaking artifacts that are present in V-BM4D and the pretrained FastDVDnet. See Appendix A.3 for full video clips. (Digital zoom recommended.)

For video-denoising, we feed in 5 noisy clips to each denoiser, then compare against a single still ground truth image. We compare our method against V-BM4D (a classic video denoising method), as well as two pretrained state of the art deep video-denoisers, FastDVDnet [200] and PaCNet [205]. Both of these models use additive Gaussian noise, so as expected, they do not perform well for our real noisy clips. FastDVDnet, which is designed to operate at multiple noise levels, outperforms PaCNet [205], which is designed

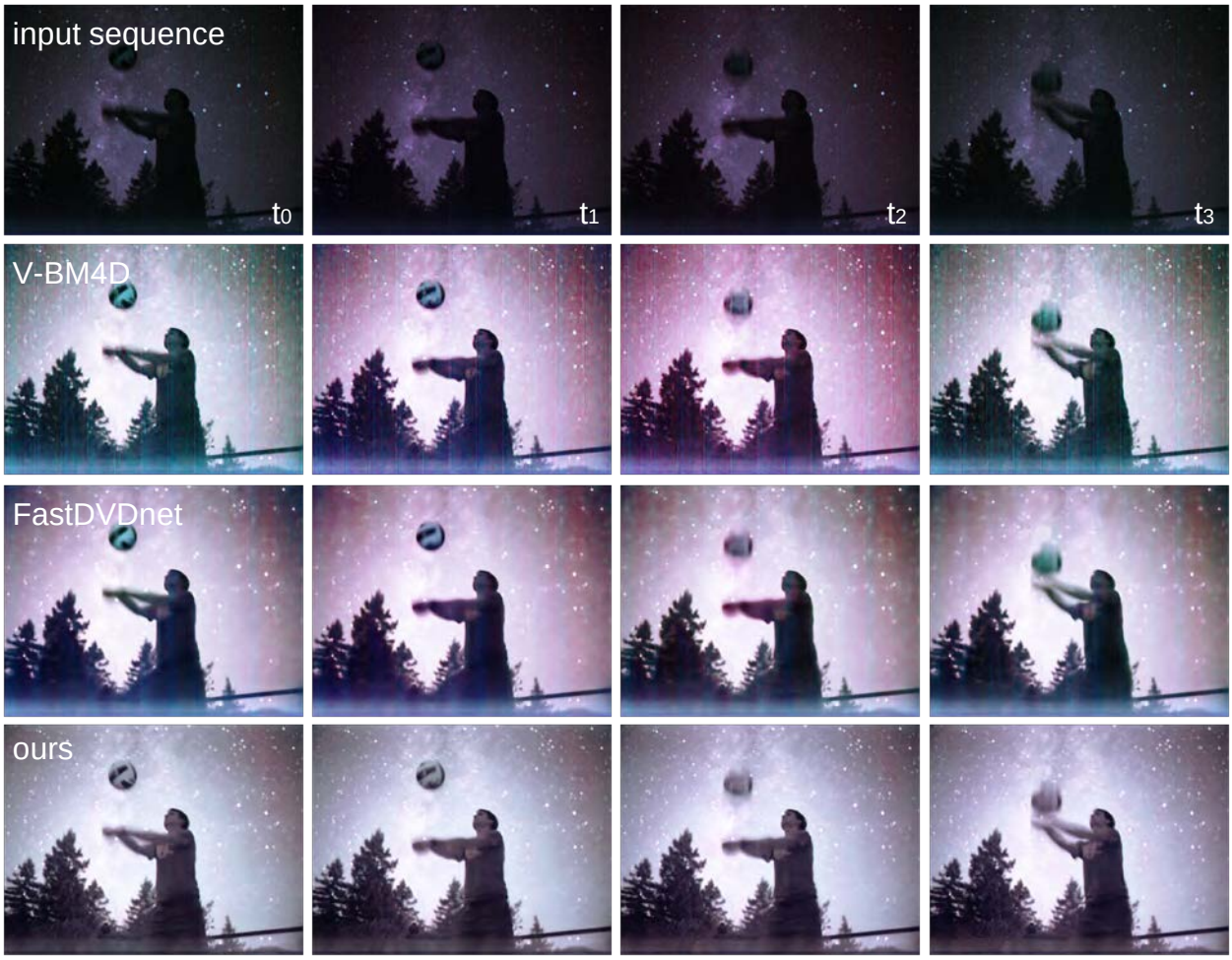


Figure 7.7: **Video Results**. Results on noisy video clip, showing performance over time for a video taken at 10fps in 0.6 mlx. Our method achieves better temporal consistency than V-BM4D or the pretrained FastDVDnet, and also has fewer artifacts within each frame. See Appendix A.3 for full video clips. (Digital zoom recommended.)

for a specific Gaussian noise level. Our denoising method, which is based on a modified FastDVDnet and trained using our noise generator, achieves the best performance. This demonstrates the importance of having a realistic noise model during denoiser training.

Next, we qualitatively compare our performance on our unlabeled dataset of submillilux video clips. We show the performance of our method as compared to V-BM4D and the pretrained FastDVDnet in Figure 7.6, with additional video comparisons available in Appendix A.3. Here we can see that our method had fewer horizontal streak artifacts than the other methods, maintains more details such as stars, and has better overall image quality. We can clearly see the milky way in the denoised videos, and our method is robust to fast-moving objects in the background (e.g. we capture a shooting star in Fig. 7.6). When viewing the ad-

jacent frames in the video clip, Fig. 7.7, we can see that our method has less flickering than V-BM4D and the pretrained FastDVDnet, which both have significant flickering between frames, likely due to the significant noise present in the raw data.

Finally, we perform a perceptual experiment with blind randomized A/B tests between our method, V-BM4D, FastDVDNet, and L2SID using 10 clips from our video dataset. Throughout 300 comparisons with 10 workers, our method is rated as having superior image quality than the alternative methods over 95% of the time (details in Appendix A.3).

## 7.6 Conclusion and Discussion

We have demonstrated photorealistic video denoising at submillilux levels of illumination for the first time. We achieved this through a combination of excellent camera hardware (a low light optimized RGB+NIR camera), a physics-inspired noise generator used to generate realistic noisy video clips, and a video trained using a combination of real still images and synthetic noisy video clip. Our work showcases the power of deep-learning-based denoising for extremely low light settings. We hope that this work leads to future scientific discoveries in extremely low light levels (e.g. studying nocturnal animal behavior in moonless conditions or under a forest canopy), and will help push the limits of robot vision and exploration in extremely dark settings. Potential misuse of this work includes night-time surveillance or use in conjunction with weapons systems.

Our approach has limitations. First, our noise generator is limited to producing noise that mimics a single gain setting (in this case, the highest gain). Future work could expand the noise generator model to work with multiple camera gains/ISOs. Next, our denoised night videos have muffled colors due to the dominance of NIR over RGB at night. Work in style transfer and recoloring could further improve the visual appearance of the denoised video clips by enhancing embedded color cues or synthesizing realistic-looking colors. Finally, the performance of the denoiser may be improved in the future through class-aware denoising [174] and joint denoising/segmentation.

# Chapter 8

# Conclusion

This dissertation explored how we can combine knowledge of imaging system physics together with deep learning to better solve imaging inverse problems. Using this strategy, we demonstrated improved performance for several different compressive computational cameras, from single-shot 3D microscopy to snapshot hyperspectral imaging and video from stills. We also demonstrated improvements for extremely low light photography, pushing cameras to their limits and showing photorealistic video denoising at submillilux levels of illumination for the first time.

We envision that the next generation of cameras, microscopes, and telescopes will be designed through the lens of computational imaging to create better, smaller, and more-capable cameras that fuel scientific discovery, art, and medicine. Physics-informed machine learning will have a role to play in making this happen. In this chapter, we focus on some existing themes, challenges, and provide suggestions for future work that we believe will be pivotal for the field of computational imaging.

## 8.1 Challenges and Open Questions

### Capturing, simulating, or generating datasets for computational cameras

As we saw throughout this dissertation, most deep learning-based approaches require large, custom datasets during the training phase. Through physics-informed machine learning, we can potentially reduce the size of the training dataset, but not entirely curb our reliance of training data (at least for supervised learning). Obtaining a custom dataset of real, aligned training pairs can be difficult, necessitate complicated and expensive hardware setups, and may be impossible to obtain for certain settings. Furthermore, a new dataset may be needed for each new computational camera, even ones with the same design, creating a complicated calibration process to get a new computational camera up and running.

As we demonstrated in Ch. 5, if we have a very well-calibrated imaging forward model, we can simulate a dataset by running existing images/volumes through our forward model or op-

tical simulator. This removes the need for an experimentally-acquired dataset of image pairs, but necessitates an accurate calibration of a camera/microscope and the ability to accurately simulate realistic measurements from said camera/microscope. For cameras/microscopes that can be described as linear systems with known forward models, this is possible and practical to do. However, for more complicated optical systems, perhaps with nonlinearities or unknown and hard to characterize forward models, this may not be feasible.

In Ch. 7, we demonstrated how we can *learn* a forward model for a computational camera using a small dataset of calibration pairs, then use this to generate a synthetic dataset. We believe this is a very powerful technique—we can model and use data to tune anything we know a priori about the camera, then pair this with a neural network that can potentially learn anything else about our system that we do not know, or that is hard to model, such as nonlinearities. In the end, we can have very expressive neural models of how our cameras/microscopes behave in extremely challenging settings, such as nonlinear regimes, or extremely high noise/high gain settings. These models can then be used to synthesize realistic training data, or perhaps be used more directly in inverse problems. We believe that this kind of physics-inspired, deep-learning-based calibration process will be useful in pushing cameras to and beyond their current limits.

## Hand-designed priors or hand-designed networks?

Unsupervised learning, as we saw in Ch. 6, is very promising for a variety of settings since it removes the need for a dataset. Instead of learning a prior or how to solve the inverse problem, the network itself can act as a prior on the imaging inverse problem without any pre-training. Having a good imaging prior is especially important for underdetermined inverse problems, so having a network architecture that serves as a good prior is critical. This brings about several questions: which network architecture makes for the best image prior? Will the optimal network architecture vary with application/task? How can we pick the right architecture for a given imaging application? These are all open questions, and are related to the age-old question in imaging of how to choose the best imaging prior. Just as there were many decades of research on hand-designed priors, such as wavelet sparsity, discrete cosine transforms, TV, dictionary learning, etc., we envision that there will be copious research on which networks serve as the best priors for given tasks.

## Robustness and Guarantees

A camera/microscope that you cannot trust is unsuitable for most scientific and clinical applications. For machine-learning-based reconstruction algorithms, knowing definitely when the algorithm works and when you can trust the reconstruction is difficult. The structure of the network and the training data used can impact the solution and potentially introduce artifacts that are indistinguishable from signal. When there is not enough training data, networks can over-fit, hallucinating information. Moreover, if there is a distribution shift between your training data and your testing data, the network can under-perform, or not generalize to the new setting.

We are hopeful that some of these challenges will be addressed by research in deep learning theory on uncertainty quantification [1, 7] and robustness. Quantifying uncertainty for deep-learning-based inverse problems is a step in the right direction [220], but needs to be taken further to provide additional guarantees and point out any potential artifacts that are caused by our reconstruction algorithms. In signal processing, many algorithms and methods have convergence guarantees and a notion of robustness. For example, compressive sensing is well studied and has guarantees based on certain conditions, such as sparsity and incoherence. We are encouraged by some of the recent work in applying signal processing theory and ideas to deep learning [192]. With a better notion of convergence, uncertainty, and robustness, we can hopefully trust our networks, even for the most important tasks, such as scientific discovery and medical imaging.

### What is the right way to combine physics with deep learning?

Throughout this dissertation, we demonstrated several different ways of building in our knowledge of the imaging system physics together with neural networks, but our exploration was by no means exhaustive, and there are many other ways of combining physics with deep learning. So, is there a right way of blending our physics models with deep networks? Are some methods better than others, or does it depend on the specific use case and application? These are all open questions, and we believe will be important research questions moving forward. In this work, most of our network architectures tended to have a physics-based layer followed by a more traditional CNN for further refinement. However, perhaps there are better, more tightly coupled ways of blending the physics together with neural networks. Furthermore, the neural network architecture is also up for debate—historically, CNNs have performed quite well, but recently they have been surpassed at many tasks by transformers [226, 39, 213]. We expect the neural network architecture choice to change through the years, and perhaps the methods used to blend physics with the deep network will change depending on the network architecture, since each network may have unique strengths and weaknesses. For example, the small convolutional kernel size of CNNs makes them bad at tackling large, full-image-sized deconvolutions (as is the case for DiffuserCam), making our Fourier-space, differentiable deconvolutional step in Ch. 5 especially useful. Transformers, or other network architectures, such as with Fourier-space convolutions, may be better suited for these problems and have different ways of best incorporating physics into them. How to best combine physics with deep learning is an open problem and perhaps has many possibilities.

## 8.2 Outlook and Future Directions

### Task-specific cameras using end-to-end learning

As machine learning advances, raw camera measurements are no longer predominantly looked at by humans, rather, the measurements are used directly for higher-level tasks (automated

pathology diagnostics, classification, 3D localization, etc.) [142, 27, 145, 154]. Despite the advances in AI, cameras are generally designed to take the most visually appealing images for humans, not to capture the most useful information for AI. By encoding additional information (e.g. depth cues, polarization, specific hyperspectral wavelengths) into camera measurements and designing algorithms to leverage this information, we can have better informed autonomous systems and enable more robust decision making.

Throughout this dissertation, we focused on differentiable reconstruction algorithms for computational cameras—however, this is only one part of the picture. With differentiable physical models for cameras, we can also treat optical design as a learnable layer within a neural network, Fig. 8.1. Then, we can end-to-end optimize the whole camera pipeline—the optics, the algorithms, together with the higher-level task. In this way, we can optimize the camera for higher-level performance (classification, segmentation, tracking, etc.) rather than for the best image quality.

As an example, certain camera parameters can be optimized, such as the lens or microlens surface profile, locations, and any aberrations (e.g. parameterized as Zernike coefficients). Light can be computationally propagated through the lens to the sensor (e.g. with Fresnel propagation) to compute the PSF of the system. This PSF can be used in the forward model, which can also include any sensor effects (e.g. Bayer filter, spectral filters) to generate simulated measurements, which can then be fed into the reconstruction algorithm and/or network controlling the higher-level task. Based on the higher-level loss, the gradients can be backpropagated through this entire network to update both the optical parameters and reconstruction parameters. Manufacturing and physical constraints can be incorporated into the optical model, making sure that the learned designs can be easily manufactured and potentially meet certain requirements, such as limits in size and weight.

We believe this concept of end-to-end design for higher-level tasks, of which physics-inspired machine learning is a key part, will be very useful in designing the next generation of computational cameras. End-to-end optical design and physics-inspired machine learning reconstructions share many common challenges and open questions. In addition to this, end-to-end design has a few unique additional challenges and considerations.

### Datasets for end-to-end design

Just as for physics-informed inverse problems, learning the optics *and* the reconstruction algorithm faces the issue of needing good datasets. However, for a given task, we do not necessarily know what is the right data to make the most informed decision. Does a robot navigating through the world benefit from having hyperspectral information for each point in the world? How about a microscope trying to classify skin cancer? When are polarization or phase information important and useful?

To find out the answers to these questions, we need to collect all of this information for real systems, needing multiple complicated imaging systems to gather the light field, polarization, depth, hyperspectral profile, and phase for coherent imaging systems. Once this is collected, we additionally need the data to be annotated for each task (e.g. segmentation). This process could be sped up through unsupervised approaches to data annotation. Alternatively,

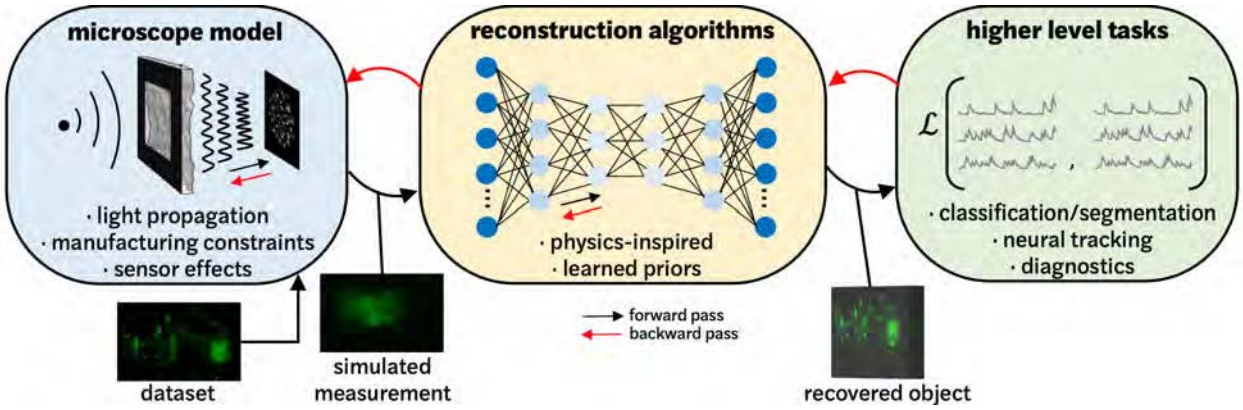


Figure 8.1: **Designing cameras for tasks: end-to-end design of imaging systems**. Through differentiable optical models and reconstruction algorithms, we can treat all aspects of the camera capture (encoding) and algorithms (decoding) process as a neural network with learnable parameters. Then, using backpropagation, we can learn the camera parameters and algorithms together to optimize a higher-level objective function, such as classification or segmentation performance. This would allow cameras to be designed to encode the most useful features from the world (e.g. different wavelengths of light, polarization, 3D, etc.) for a given task, rather than designed to take the sharpest image.

physics-based simulators could be used to train cameras based solely on synthetic data. This has shown promise in self-driving vehicle research, where high-quality physics-based world simulators synthesize realistic sensor data that is used to train algorithms for self-driving vehicles [58]. For computational imaging and microscopy, such simulators could also be used, but often have unique challenges in each domain. For example in microscopy, this sort of simulator may need to include complex light interaction, multiple-scatting, contain interesting, realistic biological samples. This is a bit of chicken and the egg problem, where we want to use our computational cameras to study and understand the world, but to design/build these cameras, it would help if we already had good models of the world and could realistically simulate the world.

## Neural representations

Over the last few years, there has been an explosion of papers and interest in using neural representations for a variety of tasks [153, 189, 152]. Rather than solving for images or volumes as a grid of pixels or voxels, with neural representations, the images/volume is the output of a neural network. We have already seen these concepts applied to imaging inverse problems [124, 235], and we expect to see these methods gain popularity for other imaging inverse problems.

# Appendix A

# Appendix

## A.1 Supplemental Materials for Supervised Learning for 2D Lensless Photography

### Network architecture

We outline our U-Net network architecture (used for Le-ADMM-U as well as for the U-Net comparison) below in Table A.1 for completeness. This is based on the architecture specified in [176]. Here **k** represents the kernel size, **s** is the stride, **channels in/out** represents the number of input and output channels for the layer, and **input** is the input of the layer, with ',' representing concatenation. Here the encoding steps, *enc*, consist of two convolutional layers, each of which consists of a 2D convolution, followed by a batch-norm and ReLU. The decoding steps, *dec*, consist of three convolutional layers with the same architecture. Here, up$(\cdot)$ stands for bilinear upsampling. *conv1* consists of a convolutional layer, batch-norm, and ReLU, whereas *conv2* consists only of a convolutional layer.

Next, we outline our smaller U-Net that is used for Le-ADMM*. The encoding and decoding steps are the same as described in Table A.1. Finally, we include a skip connection, adding the input of the network to the output. The network architecture is described as follows:

### Effect of training size

In Fig. A.1 we study the effect of the number of training images on the network performance. We show that our model-based network, Le-ADMM-U, is able to perform much better than the deep method (U-Net) with fewer training images because it incorporates knowledge of the imaging system into the network.

**Table A.1: Network architecture for U-Net used in Le-ADMM-U**

| layer | k | s | channels in/out | input |
|---|---|---|---|---|
| enc1 | 3 | 1 | 3/24 | input |
| pool1 | 2 | 2 | 24/24 | enc1 |
| enc2 | 3 | 1 | 24/64 | pool1 |
| pool2 | 2 | 2 | 64/64 | enc2 |
| enc3 | 3 | 1 | 64/128 | pool2 |
| pool3 | 2 | 2 | 128/128 | enc3 |
| enc4 | 3 | 1 | 128/256 | pool3 |
| pool4 | 2 | 2 | 256/256 | enc4 |
| enc5 | 3 | 1 | 256/512 | pool4 |
| pool5 | 2 | 2 | 512/512 | enc5 |
| conv1 | 3 | 1 | 512/512 | pool5 |
| dec5 | 3 | 1 | 512/256 | up(conv1), enc5 |
| dec4 | 3 | 1 | 256/128 | up(dec5) enc4 |
| dec3 | 3 | 1 | 128/64 | up(dec4), enc3 |
| dec2 | 3 | 1 | 64/24 | up(dec3), enc2 |
| dec1 | 3 | 1 | 24/24 | up(dec2), enc1 |
| conv2 | 1 | 1 | 24/3 | dec1 |

**Table A.2: Network architecture for U-Net used in Le-ADMM*.**

| layer | k | s | channels in/out | input |
|---|---|---|---|---|
| enc1 | 3 | 1 | 3/24 | input |
| pool1 | 2 | 2 | 24/24 | enc1 |
| conv1 | 3 | 1 | 24/24 | pool1 |
| dec1 | 3 | 1 | 24/24 | up(conv1), enc1 |
| conv2 | 1 | 1 | 24/3 | dec1 |

## A.2 Supplemental Materials for Unsupervised Learning for Compressive Lensless Photography

Throughout, our default network architecture is a 5-layer skip network, like the default network architecture used in [204]. The upsampling layer has $n_u$ filters with kernel size $k_u$, the downsampling layer has $n_d$ filters with kernel size $k_d$, and the skip layer has $n_s$ filters with kernel size $k_s$. Our default activation function (act_func) is a Leaky ReLU. At each iteration, we perturb our input with an additive normal noise $\sigma_p$, as described in [204]. We also add a small amount of regularization tv_weight to our loss before backtracking. The default parameters are given below in Table A.3.

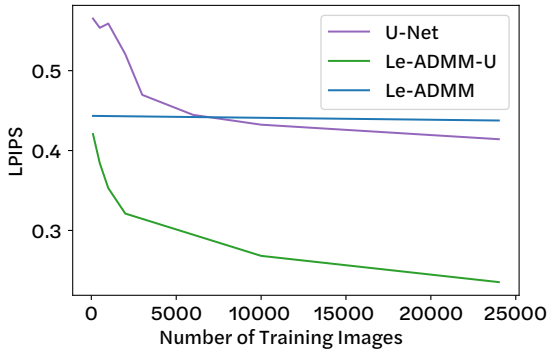


Figure A.1: **Effect of Training Size**. Here we vary the number of images in the training set and plot the LPIPS score after 5 epochs. Here we see that Le-ADMM-U performs better and converges faster than a U-Net alone. Le-ADMM does not improve as the number of training images increases, since it has so few parameters.

| |
|---|
| $n_u = n_d = [16, 32, 64, 128]$ |
| $k_u = k_d = [3, 3, 3, 3, 3]$ |
| $n_s = [4, 4, 4, 4, 4]$ |
| $\sigma_p = 0.05$ |
| LR $= 10^{-3}$ |
| tv_weight$=10^{-20}$ |
| act_func = LeakyReLU |
| upsampling = bilinear |

Table A.3: Default Network parameters

## 2D imaging with erasures

To reconstruct an RGB image of size $(3 \times W \times H)$, we feed a random input of size $(80 \times 2W \times 2H)$ to the network to get an output of size $(3 \times 2W \times 2H)$ and then crop it for the final result. The size doubling is due to our lensless forward model. The measurements are downsampled by a factor of 4 from the original measurement size, thus recovering an image with size $W = 480$ and $H = 270$. We run between $100,000 - 700,000$ iterations for the simulated data and $50,000 - 100,000$ for the experimental data. The learning rate (LR) ranges from $10^{-3}$ to $10^{-4}$ (high erasures needs a small learning rate to produce smooth reconstructions)

For experimental data, early-stopping is necessary to prevent the network from overfitting to noise in the measurement, as described in detail in [204]. An example of this early stopping can be seen in Fig. A.2.

## Single-shot video

For our simulations, we reconstruct an image of size $(3 \times 38 \times 134 \times 160)$, which corresponds to a color image for each of the 38 video frames. For this, we utilize a PSF and shutter function from [9] downsized by 16× spatially and 2× temporally. We utilize an input size of $(3 \times 38 \times 134 \times 160)$ and a 3D Skip network (with 3D convolutions instead of 2D convolutions). The parameters are the same as the defaults in Table A.3 with the exception of upsampling=trilinear, LR = 0.01, tv_weight = 0, and we run 20,000 iterations for this task. We find that 3D Skip networks outperform 2D Skip networks for certain reconstructions, however take up more GPU memory and therefore limit us to smaller reconstruction sizes.

For our experimental data, we reconstruct the video $(3 \times 72 \times 270 \times 480)$ from a still RGB image of size $(3 \times 270 \times 480)$, we run the reconstruction for each color channel separately and then combine the color channels together for the final result. The measurement is downsampled by a factor of 8 from the original measurement from [9]. For each color channel, the network input is of size $(120 \times 540 \times 960)$ with 15,000 iterations, and we utilize 2D Skip networks (with default parameters in Table 1) in order to maintain a larger reconstruction size.

## Single-shot hyperspectral

For our simulations, we reconstruct an image of size $(64 \times 150 \times 200)$, which we obtain by cropping the PSF and mask function from [158] and spatially downsampling by 2×. This corresponds to a grayscale image for each of the 64 wavelengths. We obtain our ground truth spectral image from [63] and bin the wavelengths from the dataset to match the wavelengths of the hyperspectral imager from [158]. We use input shape $(32 \times 150 \times 200)$ and a 2D Skip network with the same parameters as in Table A.3, but with a learning rate of LR = 0.01 and TV weight of 0. We plot the reconstruction after 100,000 iterations.

For our experimental data, we reconstruct an image of size $(32 \times 160 \times 160)$, which is spatially and spectrally downsampled by 2×, and cropped. We use a network input shape of $(32 \times 160 \times 160)$ and a 3D Skip network (with 3D convolutions instead of 2D convolutions) with the default network parameters, with the exception of upsampling = triliear, LR = 0.01, tv_weight = 0, and we run 4600 iterations for this task.

## Early stopping during training

As mentioned in [204], early stopping is needed to prevent overfitting to noise. In our reconstructions on experimental data, the reconstruction improves and then degrades after a certain number of iterations. Choosing an appropriate stopping point is necessary to achieve the best image quality. Figure A.2 shows an example of this on data from our 2D reconstruction with 0% erasures. Here we can see that all of our image metrics improve, and then begin to degrade after 30,000 iterations. We choose to stop the reconstruction at around 30,000 iterations, when the image has the most details (e.g. the dots in the butterfly

wings) without too much noise. As the reconstruction progresses to 80,000 iterations, more noise is visible in the image.


Figure A.2: **Early stopping for 2D experimental images**. The reconstruction gets noisier after converging to the lowest LPIPS score. In the main paper, we show results with early stopping for our experimental data; interestingly we don't see this effect on simulation data perhaps due to the lack of noise and model mismatch.

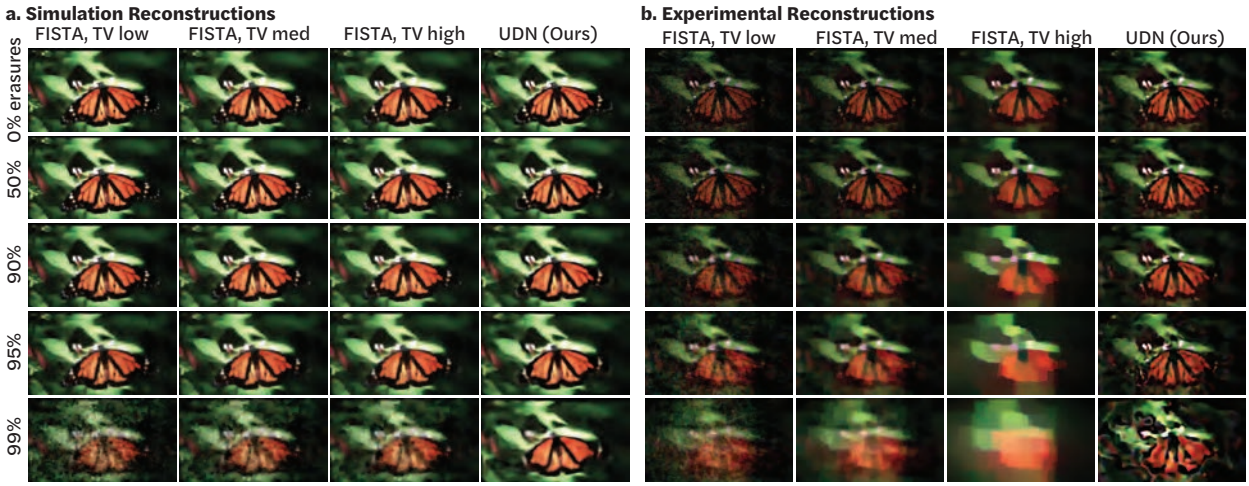


Figure A.3: **2D Compressive imaging with erasures, compared against three amounts of TV**. (a) Simulation measurements and (b) experimental measurements comparing our UDN with FISTA using three different amounts of TV. (In the main text, only the reconstruction with the best TV value is shown.) We can see that in the experimental reconstructions, too much TV tends to blur out the reconstruction while too little TV maintains noise in the reconstruction.

## Additional Reconstructions

In this section, we provide several additional reconstructions using our UDN method. First, we show our 2D erasures simulation and experimental reconstruction compared to all of the

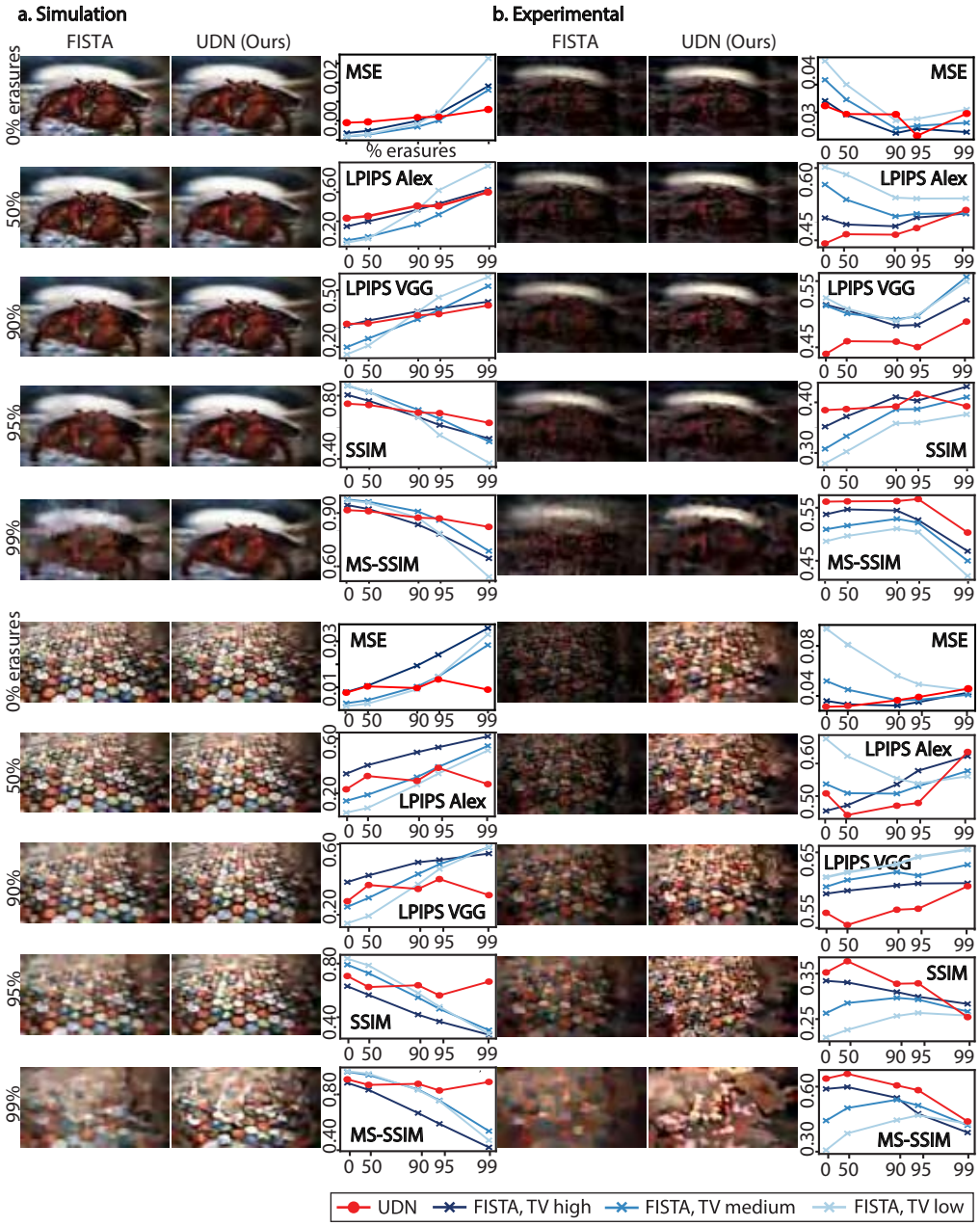


Figure A.4: **Additional 2D reconstructions with increasing percentage of erasures** for (a) simulated measurements and (b) experimental measurements. For each set of erasures, the first column shows FISTA reconstructions, the second column shows UDN reconstructions, the last column quantifies the performance against the ground truth using 5 different metrics. Generally, UDN outperforms FISTA when there are more erasures.

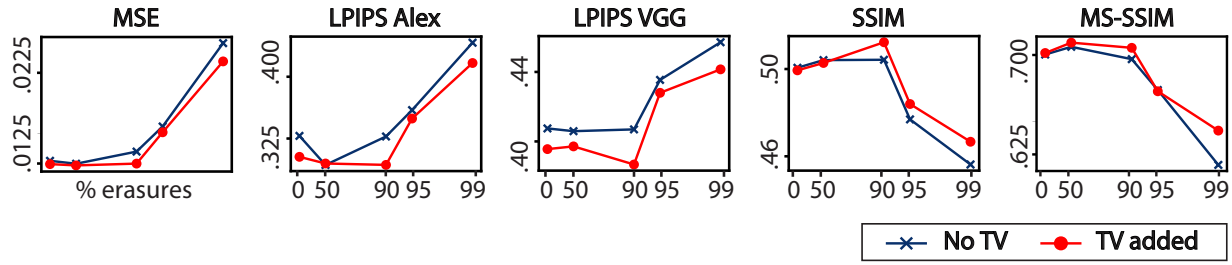


Figure A.5: **2D imaging with erasures, with and without TV regularization**. We compare the 2D UDN experimental reconstructions at 0%, 50%, 90%, 95%, 99% erasures with and without TV. We find that the UDN serves as an effective prior alone, but a small amount of TV consistently provides a slight improvement in image quality.

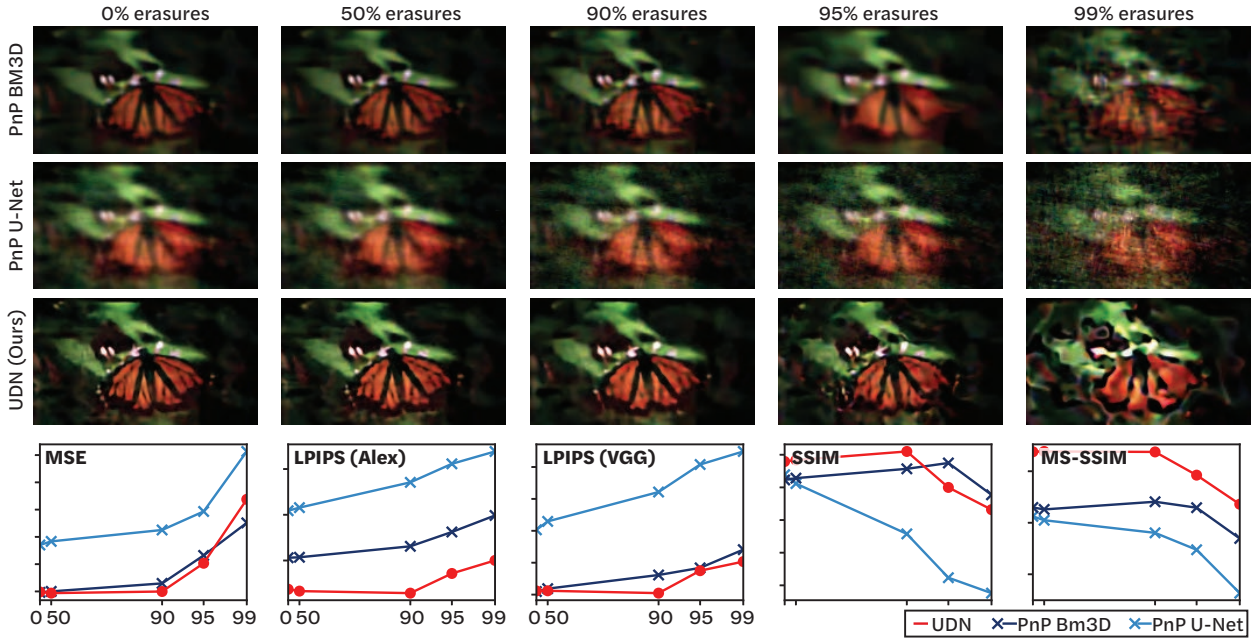


Figure A.6: **2D Compressive imaging with erasures, compared against Plug and Play Priors**. Experimental reconstructions comparing our UDN with FISTA using two different Plug and Play Priors (PnP): BM3D and a pre-trained U-Net. Here we can see that PnP BM3D does quite well at higher erasures, but our UDN is able to maintain higher frequency features and outperforms PnP BM3D on most perceptual metrics. PnP U-Net has worse image quality than either of the other two methods.

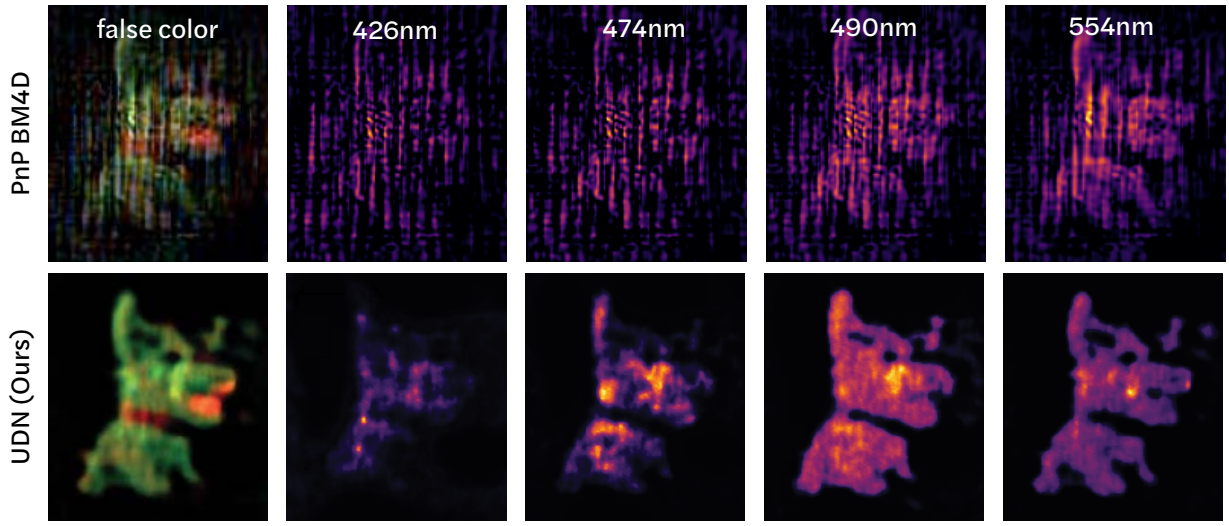


Figure A.7: **Comparison against Plug and Play Priors on hyperspectral data**. Experimental reconstructions comparing our UDN with PnP BM4D. Here we can see that PnP BM4D is not able to adequately regularize our hyperspectral data, resulting in prominent reconstruction artifacts (stripes) across the image.

FISTA levels, as opposed to only the best TV amount according to the LPIPS-Alex metric, as shown in the paper, Fig. A.3. Here we can see that TV becomes ineffective for 99% erasures both in simulation and experiment for all levels of regularization. Next, we show reconstructions of two additional 2D measurements with erasures, Fig. A.4. As before, we see a similar trend in the simulated reconstructions, showing that UDN provides a better prior on the image reconstruction for extreme erasures. On the experimental data, UDN appears to provide a better prior with a better LPIPS-Alex and LPIPS-VGG score for all levels of erasure.

Adding a small amount of TV regularization to the UDN can improve network performance, an observation that was studied in detail in [133]. We compare our UDN's performance with and without TV on the task of 2D imaging with erasures in experiment, shown in Fig. A.5. We see that the network alone serves as an effective prior, but adding TV can provide a slight enhancement; this matches the observations of [133].

We also compare UDN against two Plug and Play Priors (PnP): BM3D and a pre-trained U-Net [230] for image denoising, Fig. A.6. We can see that PnP BM3D provides a slightly better prior for 2D imaging with erasures, but is outperformed by UDN for perceptual image quality and ability to maintain high frequency content. PnP U-Net does not perform very well, likely due to the fact that it's pre-trained for additive white Gaussian noise, and does not generalize very well our imaging system and to reconstruction artifacts. In addition, we compare against PnP BM4D for hyperspectral imaging in Fig. A.7. Due to the computational complexity and lack of GPU acceleration for BM4D, we spatially and spectrally downsize by 2×. Overall, we can see that PnP BM4D is not able to produce as

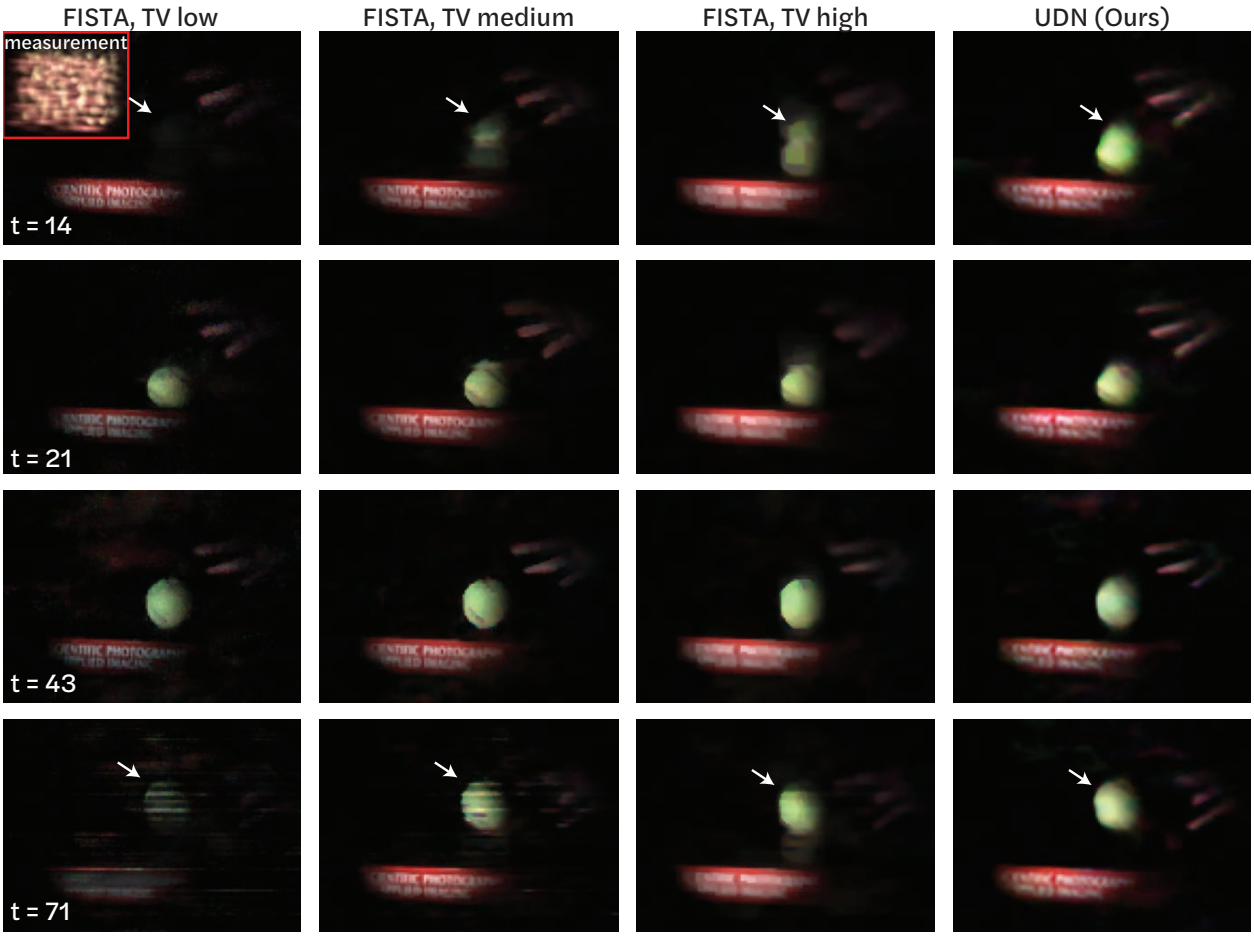


Figure A.8: **Additional experimental results on rolling shutter data**. Here we see that our reconstruction better captures the motion in the video (e.g. the ball movement), whereas the FISTA reconstruction has noticeable artifacts and the ball even disappears for lower TV values. See Visualization 3 for the full video.

clean of a reconstruction as UDN and has prominent artifacts visible in the reconstruction. At our reduced resolution, PnP BM4D takes around 24 hours for a reconstruction, making tuning this algorithm difficult. Performing the reconstruction at full resolution would take multiple days, which is not practical, and notably slower than our UDN reconstruction.

Finally, we show another single-shot video experimental result in Fig. A.8. We can see that UDN provides more uniform image quality throughout the video, and also is better at capturing the scene motion. FISTA has more noticeable artifacts, however is able to recover slightly sharper features (e.g. the book text) at the cost of more artifacts throughout the video.

## A.3 Supplemental Materials for Learning a Physics-informed Noise Model

### Additional implementation details

#### Noise Generator and Discriminator

First, we provide additional details about the noise generator and discriminator. In our noise generator, we include a periodic noise component. This periodic noise component is modeled as follows in the frequency domain as:

$$n[M,N] = \mathcal{F}^{-1}\begin{cases} X_1, & \text{if } N = 0 \\ X_2 + X_3 j, & N = N_t/4 \\ X_2 - X_3 j, & N = 3N_t/4 \\ 0, & \text{otherwise} \end{cases}$$

Where $X_1$, $X_2$, and $X_3$ are zero mean Gaussian random variables with optimized variances $\lambda_{f1}$, $\lambda_{f2}$, $\lambda_{f3}$, $N_t$ is the total number of columns in the image, and $M, N$ index the rows/columns of the image. This essentially corresponds to adding a 1 or 2 pixel period sinusoidal pattern to the image with a random amplitude that is determined by the optimized variance parameter. We demonstrate the effect of this noise in Figure A.9, showing a central slice of the Fourier transform of the clean vs. noisy images. We can see that our full model better matches the real noise than our partial model which does not consider the periodic noise component.

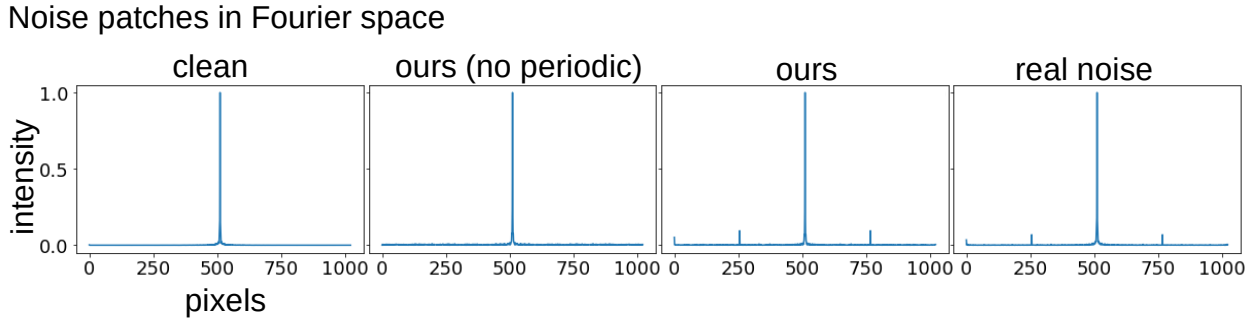


Figure A.9: **Fourier transform of the noise**. Fourier transform of clean and noisy patches, showing the prominent spike in Fourier space that we see in the real noisy images. Our full model captures this behaviour.

For the CNN in our noise generator, we use a standard 2D residual U-Net architecture, with 4 input and output channels, 4 upsampling and downsampling layers, stride-2 convo-

lutional downsampling layers, stride-2 transpose convolutional upsampling layers, and SeLU activations. The number of channels in our 4 downsampling and upsampling layers are 32, 64, 128, and 256.

We initialize our shot, read, row, and row temporal noise parameters to 0.2, 0.02, and 0.002, and 0.0002 respectively. We initialize our uniform noise parameter to 0.1, and our periodic parameters to 5. In general, we note that the initialized value of these parameters did not seem to effect the final converged value as long as the initial values were small.

Our discriminator's architecture is outlined in Fig. A.10. We feed in images with a patch size of 64 into the discriminator during training. We use an Adam optimizer [106] with a learning rate of 0.0002, with $\beta_1 = 0.5$ and $b_2 = 0.999$ for both the generator and discriminator. For each experiment in our generator ablation study, we feed both the noisy patch as well as the Fourier transform of the noisy patch into the discriminator, which we found resulted in better performance than using either the image or the Fourier transform of the image alone. For the final two comparisons in the ablation (all the noise parameters with U-Net and all the noise parameters without the U-Net) we use only the Fourier transform of the image in the discriminator, which resulted in the best performance given those parameters. In all experiments, we add an LPIPS loss to our generator loss. We take a gradient step on the generator after every 5 gradient steps on the discriminator.

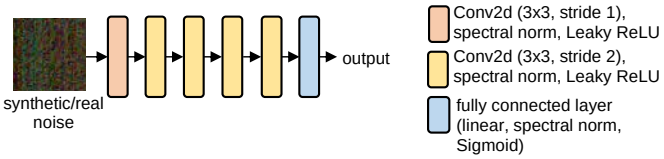


Figure A.10: **Discriminator architecture**. We use our discriminator during our noise generator training.

### Denoiser details

We base our denoiser on FastDVDnet [200]. We modify the FastDVDnet architecture in two ways. First, we increase the number of channels to 4 instead of 3 to facilitate processing our RAW images. Next, we modify the denoiser blocks. The original implementation of FastDVDnet uses a U-Net architecture for the denoising blocks. We replace this architecture with HRNet blocks. In our raw high gain, low light videos, we often see flashing and differences in colors between frames (Figure A.11). Experimentally, we found that using HRNet blocks reduces the flickering across frames that we see at our lowest light settings. Figure A.11 shows an example of this with a FastDVDNet denoiser using U-Net blocks vs. as FastDVDnet denoising with HRNet blocks. When plotting the mean intensity over time, we can see that version with HRNet blocks has less variance in the intensity, effectively smoothing out

the flickering over frames, whereas the FastDVDNet with U-Net blocks is not effective at reducing flickering, having a higher variance in the mean intensity over time.

Following from FastDVDnet, our denoiser architecture consists of two denoising blocks. Each block takes in 3 images with 4 channels each (12 channels total) and outputs a single image with 4 channels. We use an HRNet designed for semantic segmentation [194, 211, 225] and slightly modify it to work on our images by replacing the initial stride-2 convolutions to stride-1 convolutions. Our HRNet has 4 stages. The first stage consists of a Bottleneck block, while the remaining stages consists of Basic blocks. We summarize the number of branches ($N_{br}$), number of channels ($N_c$), number of modules ($N_m$), and number of blocks ($N_{bl}$) in each stage in the Table A.4.

| Stage | $N_{br}$ | $N_c$ | $N_m$ | $N_{bl}$ |
|---|---|---|---|---|
| 1 | 1 | [64] | 1 | [4] |
| 2 | 2 | [18, 36] | 1 | [4,4] |
| 3 | 3 | [18, 36, 72] | 3 | [4,4,4] |
| 4 | 4 | [18, 36, 72, 144] | 3 | [4,4,4,4] |

Table A.4: HRNet architecture

### Camera details

For all noisy sequences, we use the highest camera gain: 16× column amplifier gain), 6dB CDS Gain, and 1023 VGA gain. In addition, all images are stored as RAW unprocessed images with RGB+NIR channels. The clean images were taken at 1× column amplifier gain, 6dB CDS Gain, and 10 VGA gain. For all images, the exposure on the clean/noisy images was set to approximately match their intensities ( 1000×). For all sets of images, a scalar offset to correlate the mean intensity for the clean/noisy pairs was calculated and applied to each clean image to match the mean to that of the noisy images.

## Additional Results

### Simplified Noise Model results

Next, we perform an additional ablation in which we keep our denoiser constant and use a different noise generator to synthesize our noisy video clips. We compare using our full noise generator against our full noise generator without the U-Net and with only read, shot, and uniform noise in Table A.5, showing the performance on our stills dataset. We can see that our full model with the U-Net performs the best. We anticipate that the U-Net is able to learn additional features of the noise that we do not explicitly model (e.g. chromatic effects in the noise) and perhaps augment any simplifications in our noise mode (e.g. using a heteroscedastic Gaussian noise model rather than a Poisson model for shot noise). Given

better synthetic noise, our denoiser can successfully tackle sensor-specific noise and produce the best denoised images.

| Method | PSNR | SSIM | LPIPS |
|---|---|---|---|
| Ours (Shot+read+uniform) | 23.8 | 0.861 | 0.111 |
| Ours no U-Net | 25.5 | 0.910 | 0.115 |
| **ours** (full) | **27.7** | **0.931** | **0.078** |

Table A.5: Performance on still images from test set.

### Stills Results

Next we provide additional images to showcase our results on our stills dataset. We compare against V-BM4D, L2SID, N2S, and FastDVDNet. We compare both against pretrained L2SID as well as L2SID retrained using our stills dataset. Two example images are shown in Figure A.12. Here we can see that V-BM4D and N2S both have significant line artifacts throughout the images. Pretrained L2SID has issues with color, since our camera has an additional NIR channel rather than only RGB channels, and also blurs out features due to differences in our camera noise. When we pretrain L2SID using our own data, the performance improves substantially for still images as expected. Since L2SID is a single-image method and ours uses 5 images to collaboratively denoise an image, we still outperform L2SID in dark regions of the image (e.g. the text on the cans in clip 2). Furthermore, retrained L2SID results in severe flickering and poor performance on moving videos (see attached video clips). Similarly, pretrained FastDVDNet contains stripe artifacts and has reduced resolution since it is trained using a Gaussian noise model. Our method closely matches the ground truth images, maintaining the image features while suppressing the noise.

See attached supplemental video for a video comparison between our method, V-BM4D, L2SID (retrained on our stills data), and FastDVDNet. All videos are downsized by 2× from the full resolution and cropped by 880×630 pixels (full resolution is 2160×1280). In addition, we provide a video with a compilation of denoised clips from our submillilux dataset. In these videos, we demonstrate the performance of our denoiser at the most challenging low light setting with significant motion.

### Perceptual Experiments

We perform a perceptual experiment with blind randomized A/B tests between our method, V-BM4D, FastDVDNet, and L2SID. We show 10 clips from our submillilux dataset. Each clip is 30 frames long and is cropped to a 400×400 region which shows significant motion. During the experiment, we show 2 video clips side by side in a randomized order and workers are asked which clip they prefer. We run 300 comparisons in total with 10 workers. The results are summarized below:

- 95.0 [+/- 4.27]% prefer our method over FastDVDNet.

- 99.0 [+/- 1.95]% prefer our method over L2SID.
- 97.0 [+/- 3.34]% prefer our method over V-BM4D.

As we can see, in all the experiments, video clips produced by our method are preferred over alternative methods by a large margin.

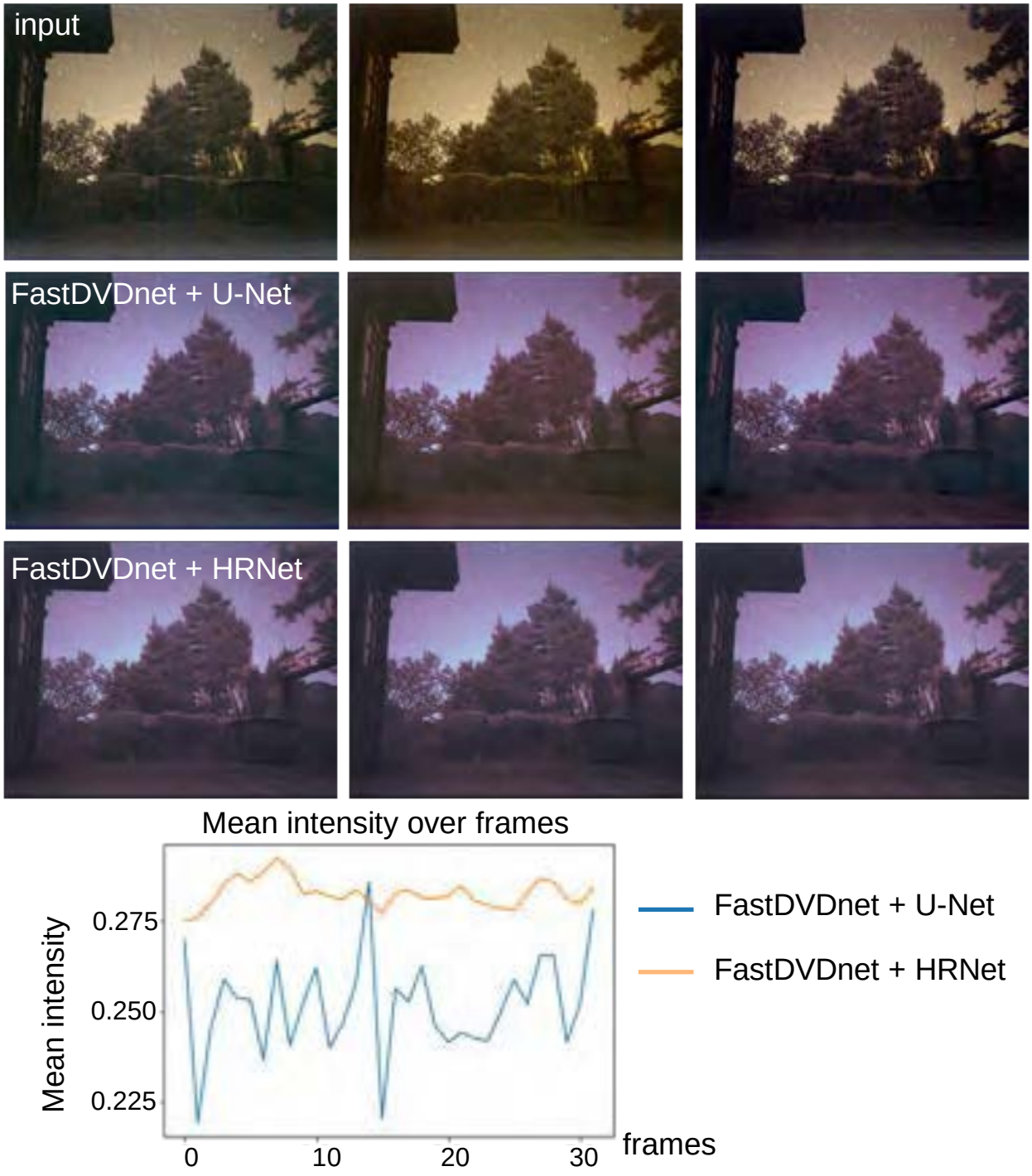


Figure A.11: **Architecture comparison**: FastDVDnet + U-Net vs FastDVDnet + HRnet. Here we can see that original FastDVDnet results in more flickering between frames than our modified FastDVDnet (with HRnet).


Figure A.12: **Denoising comparison** on two noisy bursts of still objects from our test set.

# Bibliography


[1] Moloud Abdar et al. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges". In: *Information Fusion* 76 (2021), pp. 243–297.

[2] Abdelrahman Abdelhamed, Marcus A Brubaker, and Michael S Brown. "Noise flow: Noise modeling with conditional normalizing flows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3165–3173.

[3] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. "A high-quality denoising dataset for smartphone cameras". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1692–1700.

[4] Jesse K Adams et al. "Single-frame 3D fluorescence microscopy with ultraminiature lensless FlatScope". In: *Science advances* 3.12 (2017), e1701548.

[5] Hamed Akbari et al. "Hyperspectral imaging and quantitative analysis for prostate cancer detection". In: *Journal of Biomedical Optics* 17.7 (2012), p. 076005.

[6] Josue Anaya and Adrian Barbu. "Renoir–a dataset for real low-light image noise reduction". In: *Journal of Visual Communication and Image Representation* 51 (2018), pp. 144–154.

[7] Anastasios N Angelopoulos et al. "Image-to-image regression with distribution-free uncertainty quantification and applications in imaging". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 717–730.

[8] Nick Antipa et al. "DiffuserCam: lensless single-exposure 3D imaging". In: *Optica* 5.1 (2018), pp. 1–9.

[9] Nick Antipa et al. "Video from Stills: Lensless imaging with rolling shutter". In: *2019 IEEE International Conference on Computational Photography (ICCP)*. IEEE. 2019, pp. 1–8.

[10] Muthuvel Arigovindan et al. "A Parallel Product-Convolution approach for representing depth varying Point Spread Functions in 3D widefield microscopy based on principal component analysis". In: *Opt. Express* 18.7 (Mar. 2010), pp. 6461–6476. DOI: 10.1364/OE.18.006461. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-18-7-6461.

[11] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[12] Salim Arslan et al. "Attributed relational graphs for cell nucleus segmentation in fluorescence microscopy images". In: *IEEE transactions on medical imaging* 32.6 (2013), pp. 1121–1131.

[13] Gaurav Arya et al. "End-to-End Optimization of Metasurfaces for Imaging with Compressed Sensing". In: *arXiv preprint arXiv:2201.12348* (2022).

[14] M Salman Asif et al. "FlatCam: Replacing lenses with masks and computation". In: *Computer Vision Workshop (ICCVW), 2015 IEEE International Conference on*. IEEE. 2015, pp. 663–666.

[15] M Salman Asif et al. "Flatcam: Thin, lensless cameras using coded aperture and computation". In: *IEEE Transactions on Computational Imaging* 3.3 (2016), pp. 384–397.

[16] European Machine Vision Association. "EMVA standard 1288: Standard for characterization of image sensors and cameras". In: (2012).

[17] Christina P Bacon, Yvette Mattley, and Ronald DeFrece. "Miniature spectroscopic instrumentation: applications to biology and chemistry". In: *Review of Scientific instruments* 75.1 (2004), pp. 1–16.

[18] Seung-Hwan Baek et al. "Compact single-shot hyperspectral imaging using a prism". In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), pp. 1–12.

[19] Joshua Batson and Loic Royer. "Noise2self: Blind denoising by self-supervision". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 524–533.

[20] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202.

[21] Saima Ben Hadj, Laure Blanc-Féraud, and Gilles Aubert. "Space variant blind image restoration". In: *SIAM Journal on Imaging Sciences* 7.4 (2014), pp. 2196–2225.

[22] David SC Biggs. "3D deconvolution microscopy". In: *Current Protocols in Cytometry* 52.1 (2010), pp. 12–19.

[23] Mathieu Blondel et al. "Efficient and modular implicit differentiation". In: *arXiv preprint arXiv:2105.15183* (2021).

[24] Ashish Bora et al. "Compressed sensing using generative models". In: *arXiv preprint arXiv:1703.03208* (2017).

[25] Emrah Bostan et al. "Deep phase decoder: self-calibrating phase microscopy with an untrained deep neural network". In: *Optica* 7.6 (2020), pp. 559–562.

[26] Assim Boukhayma. "Low-noise CMOS image sensors". In: *Ultra Low Noise CMOS Image Sensors*. Springer, 2018, pp. 13–34.

[27] Nicholas Boyd et al. "Deeploco: Fast 3d localization microscopy using neural networks". In: *BioRxiv* (2018), p. 267096.

[28] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends in Machine learning* 3.1 (2011), pp. 1–122.

[29] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[30] Tim Brooks et al. "Unprocessing images for learned raw denoising". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11036–11045.

[31] Antoni Buades, Bartomeu Coll, and J-M Morel. "A non-local algorithm for image denoising". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. IEEE. 2005, pp. 60–65.

[32] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. "A review of image denoising algorithms, with a new one". In: *Multiscale modeling & simulation* 4.2 (2005), pp. 490–530.

[33] Harold C Burger, Christian J Schuler, and Stefan Harmeling. "Image denoising: Can plain neural networks compete with BM3D?" In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 2392–2399.

[34] Edmund Butt. *Night on Earth*. Plimsoll Productions. 2020.

[35] Emmanuel J Candès. "Compressive sampling". In: *Proceedings of the international congress of mathematicians*. Vol. 3. European Mathematical Society. 2006, pp. 1433–1452.

[36] Emmanuel J Candès and Carlos Fernandez-Granda. "Towards a mathematical theory of super-resolution". In: *Communications on Pure and Applied Mathematics* 67.6 (2014), pp. 906–956.

[37] Emmanuel J Candès and Michael B Wakin. "An introduction to compressive sampling". In: *IEEE signal processing magazine* 25.2 (2008), pp. 21–30.

[38] Xun Cao et al. "Computational snapshot multispectral cameras: Toward dynamic capture of the spectral world". In: *IEEE Signal Processing Magazine* 33.5 (2016), pp. 95–108.

[39] Nicolas Carion et al. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[40] Centre for Biomedical Image Analysis. *CytoPacq*. (`https://cbia.fi.muni.cz/simulator`).

[41] Maumita Chakrabarti, Michael Linde Jakobsen, and Steen G Hanson. "Speckle-based spectrometer". In: *Optics Letters* 40.14 (2015), pp. 3264–3267.

[42] Ke-Chi Chang et al. "Learning Camera-Aware Noise Models". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2020.

[43] Kuanglin Chao et al. "Hyperspectral-multispectral line-scan imaging system for automated poultry carcass inspection applications for food safety". In: *Poultry Science* 86.11 (2007), pp. 2450–2460.

[44] Chen Chen et al. "Learning to see in the dark". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3291–3300.

[45] Chen Chen et al. "Seeing motion in the dark". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3185–3194.

[46] Jingwen Chen et al. "Image blind denoising with generative adversarial network based noise modeling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3155–3164.

[47] Sunghyun Cho and Seungyong Lee. "Fast motion deblurring". In: *ACM SIGGRAPH Asia 2009 papers*. 2009, pp. 1–8.

[48] Michele Claus and Jan van Gemert. "Videnn: Deep blind video denoising". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[49] Jianan Cui et al. "PET image denoising using unsupervised deep learning". In: *European journal of nuclear medicine and molecular imaging* 46.13 (2019), pp. 2780–2789.

[50] Kostadin Dabov et al. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.

[51] Kostadin Dabov et al. "Image denoising with block-matching and 3D filtering". In: *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*. Vol. 6064. International Society for Optics and Photonics. 2006, p. 606414.

[52] Ingrid Daubechies, Michel Defrise, and Christine De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint". In: *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences* 57.11 (2004), pp. 1413–1457.

[53] Stephanie Delalieux et al. "Hyperspectral reflectance and fluorescence imaging to detect scab induced stress in apple leaves". In: *Remote Sensing* 1.4 (2009), pp. 858–874.

[54] Loic Denis et al. "Fast approximations of shift-variant blur". In: *International Journal of Computer Vision* 115.3 (2015), pp. 253–278.

[55] Steven Diamond et al. "Dirty pixels: Optimizing image classification architectures for raw sensor data". In: *arXiv preprint arXiv:1701.06487* (2017).

[56] Steven Diamond et al. "Unrolled optimization with deep priors". In: *arXiv preprint arXiv:1705.08041* (2017).

[57] Weisheng Dong et al. "Image restoration via simultaneous sparse coding: Where structured sparsity meets gaussian scale mixture". In: *International Journal of Computer Vision* 114.2 (2015), pp. 217–232.

[58] Alexey Dosovitskiy et al. "CARLA: An open urban driving simulator". In: *Conference on robot learning*. PMLR. 2017, pp. 1–16.

[59] Marco F Duarte et al. "Single-pixel imaging via compressive sampling". In: *IEEE signal processing magazine* 25.2 (2008), pp. 83–91.

[60] Thibaud Ehret et al. "Model-blind video denoising via frame-to-frame training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11369–11378.

[61] Michael Elad and Michal Aharon. "Image denoising via sparse and redundant representations over learned dictionaries". In: *IEEE Transactions on Image processing* 15.12 (2006), pp. 3736–3745.

[62] Shay Elmalem and Raja Giryes. "A Lensless Polarization Camera". In: *Computational Optical Sensing and Imaging*. Optica Publishing Group. 2021, CTh7A–1.

[63] Robert Ennis et al. "Hyperspectral database of fruits and vegetables". In: *JOSA A* 35.4 (2018), B256–B266.

[64] Rob Fergus, Antonio Torralba, and William T. Freeman. "Random Lens Imaging". In: MIT CSAIL Technical Report 2006-058, 2006.

[65] DA Fish et al. "Blind deconvolution by means of the Richardson–Lucy algorithm". In: *JOSA A* 12.1 (1995), pp. 58–65.

[66] Ralf C Flicker and François J Rigaut. "Anisoplanatic deconvolution of adaptive optics images". In: *JOSA A* 22.3 (2005), pp. 504–513.

[67] Alessandro Foi. "Clipped noisy images: Heteroskedastic modeling and practical denoising". In: *Signal Processing* 89.12 (2009), pp. 2609–2629.

[68] Alessandro Foi et al. "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data". In: *IEEE Transactions on Image Processing* 17.10 (2008), pp. 1737–1754.

[69] Rebecca French, Sylvain Gigan, and Otto L Muskens. "Speckle-based hyperspectral imaging combining multiple scattering and compressive sensing in nanowire mats". In: *Optics Letters* 42.9 (2017), pp. 1820–1823.

[70] Liang Gao et al. "Single-shot compressed ultrafast photography at one hundred billion frames per second". In: *Nature* 516.7529 (2014), pp. 74–77.

[71] Nahum Gat. "Imaging spectroscopy using tunable filters: a review". In: *Wavelet Applications VII*. Vol. 4056. International Society for Optics and Photonics. 2000, pp. 50–64.

[72] Michael E Gehm et al. "Single-shot compressive spectral imaging with a dual-disperser architecture". In: *Optics express* 15.21 (2007), pp. 14013–14027.

[73] Patrick R Gill et al. "Thermal Escher Sensors: Pixel-efficient Lensless Imagers Based on Tiled Optics". In: *Computational Optical Sensing and Imaging*. Optical Society of America. 2017, CTu3B–3.

[74] Clément Godard, Kevin Matzen, and Matt Uyttendaele. "Deep burst denoising". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 538–554.

[75] Michael A Golub et al. "Compressed sensing snapshot spectral imaging by a regular digital camera with an added optical diffuser". In: *Applied Optics* 55.3 (2016), pp. 432–443.

[76] Kuang Gong et al. "PET image reconstruction using deep image prior". In: *IEEE transactions on medical imaging* 38.7 (2018), pp. 1655–1665.

[77] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *NIPS*. 2014.

[78] AA Gowen et al. "Hyperspectral imaging–an emerging process analytical tool for food quality and safety control". In: *Trends in Food Science & Technology* 18.12 (2007), pp. 590–598.

[79] Robert O Green et al. "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)". In: *Remote Sensing of Environment* 65.3 (1998), pp. 227–248.

[80] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress. 2010, pp. 399–406.

[81] Jiuxiang Gu et al. "Recent advances in convolutional neural networks". In: *Pattern recognition* 77 (2018), pp. 354–377.

[82] Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5767–5777. URL: https://proceedings.neurips.cc/paper/2017/hash/892c3b1c6dccd52936e27cbd0ff683d6-Abstract.html.

[83] Per Christian Hansen, James G Nagy, and Dianne P O'leary. *Deblurring images: matrices, spectra, and filtering*. SIAM, 2006.

[84] Samuel W Hasinoff. *Photon, Poisson Noise*. 2014.

[85] Samuel W Hasinoff et al. "Burst photography for high dynamic range and low-light imaging on mobile cameras". In: *ACM Transactions on Graphics (ToG)* 35.6 (2016), pp. 1–12.

[86] Jonathan Hauser et al. "Dual-camera snapshot spectral imaging with a pupil-domain optical diffuser and compressed sensing algorithms". In: *Applied Optics* 59.4 (2020), pp. 1058–1070.

[87] Eugene Hecht. *Optics*. Pearson Education India, 2012.

[88] Reinhard Heckel and Paul Hand. "Deep decoder: Concise image representations from untrained non-convolutional networks". In: *arXiv preprint arXiv:1810.03982* (2018).

[89] Andrew Hennessy, Kenneth Clarke, and Megan Lewis. "Hyperspectral Classification of Plants: A Review of Waveband Selection Generalisability". In: *Remote Sensing* 12.1 (2020), p. 113.

[90] Yasunobu Hitomi et al. "Video from a single coded exposure photograph using a learned over-complete dictionary". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 287–294.

[91] Ryoichi Horisaki et al. "Three-Dimensional Information Acquisition Using a Compound Imaging System". In: *Optical Review* 14.5 (2007), pp. 347–350. ISSN: 1349-9432. DOI: 10.1007/s10043-007-0347-z. URL: http://dx.doi.org/10.1007/s10043-007-0347-z.

[92] Gang Huang et al. "Lensless imaging by compressive sensing". In: *2013 IEEE International Conference on Image Processing*. IEEE. 2013, pp. 2101–2105.

[93] Wenqian Huang et al. "Development of a multispectral imaging system for online detection of bruises on apples". In: *Journal of Food Engineering* 146 (2015), pp. 62–71.

[94] Mark J. Huiskes and Michael S. Lew. "The MIR Flickr Retrieval Evaluation". In: *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*. Vancouver, Canada: ACM, 2008.

[95] Daniel S. Jeon et al. "Compact Snapshot Hyperspectral Imaging with Diffracted Rotation". In: *ACM Trans. Graph.* 38.4 (July 2019). ISSN: 0730-0301. DOI: 10.1145/3306346.3322946. URL: https://doi.org/10.1145/3306346.3322946.

[96] Haiyang Jiang and Yinqiang Zheng. "Learning to see moving objects in the dark". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7324–7333.

[97] Kyong Hwan Jin et al. "Deep convolutional neural network for inverse problems in imaging". In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522.

[98] Kyong Hwan Jin et al. "Time-Dependent Deep Image Prior for Dynamic MRI". In: *arXiv preprint arXiv:1910.01684* (2019).

[99] K.Tajima et al. "Lensless light-field imaging with multi-phased fresnel zone aperture". In: *2017 IEEE International Conference on Computational Photography (ICCP)*. May 2017, pp. 76–82.

[100] Ulugbek S Kamilov. "A parallel proximal algorithm for anisotropic total variation minimization". In: *IEEE Transactions on Image Processing* 26.2 (2016), pp. 539–548.

[101] Almut Kelber, Anna Balkenius, and Eric J Warrant. "Scotopic colour vision in nocturnal hawkmoths". In: *Nature* 419.6910 (2002), pp. 922–925.

[102] Michael Kellman et al. "Memory-efficient learning for large-scale computational imaging". In: *IEEE Transactions on Computational Imaging* 6 (2020), pp. 1403–1414.

[103] Michael R Kellman et al. "Physics-based learned design: optimized coded-illumination for quantitative phase imaging". In: *IEEE Transactions on Computational Imaging* 5.3 (2019), pp. 344–353.

[104] Robert T Kester et al. "Real-time snapshot hyperspectral imaging endoscope". In: *Journal of Biomedical Optics* 16.5 (2011), p. 056005.

[105] Salman Siddique Khan et al. "Flatnet: Towards photorealistic scene reconstruction from lensless measurements". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[106] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[107] Amit Kohli et al. "Linear Revolution-Invariance: Modeling and Deblurring Spatially-Varying Imaging Systems". In: *arXiv preprint arXiv:2206.08928* (2022).

[108] Mikhail Konnik and James Welsh. "High-level numerical simulations of noise in CCD and CMOS photosensors: review and tutorial". In: *arXiv preprint arXiv:1412.4031* (2014).

[109] Can Fahrettin Koyuncu, Rengul Cetin-Atalay, and Cigdem Gunduz-Demir. "Object-Oriented Segmentation of Cell Nuclei in Fluorescence Microscopy Images". In: *Cytometry Part A* 93.10 (2018), pp. 1019–1028.

[110] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. "Noise2void-learning denoising from single noisy images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2129–2137.

[111] Alexander Krull et al. "Probabilistic noise2void: Unsupervised content-aware denoising". In: *Frontiers in Computer Science* 2 (2020), p. 5.

[112] Grace Kuo et al. "3D Fluorescence Microscopy with DiffuserCam". In: *Computational Optical Sensing and Imaging*. Optical Society of America. 2018, CM3E–3.

[113] Grace Kuo et al. "DiffuserCam: diffuser-based lensless cameras". In: *Computational Optical Sensing and Imaging*. Optical Society of America. 2017, CTu3B–2.

[114] Grace Kuo et al. "On-chip fluorescence microscopy with a random microlens diffuser". In: *Optics Express* 28.6 (2020), pp. 8384–8399.

[115] Grace Kuo et al. "Spatially-varying microscope calibration from unstructured sparse inputs". In: *Computational Optical Sensing and Imaging*. Optical Society of America. 2020, CF4C–4.

[116] Pierre-Jean Lapray et al. "Multispectral filter arrays: Recent advances and practical implementation". In: *Sensors* 14.11 (2014), pp. 21626–21659.

[117] Laura Leal-Taixé et al. "Motchallenge 2015: Towards a benchmark for multi-target tracking". In: *arXiv preprint arXiv:1504.01942* (2015).

[118] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. "A nonlocal Bayesian image denoising algorithm". In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1665–1688.

[119] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[120] Jaakko Lehtinen et al. "Noise2Noise: Learning Image Restoration without Clean Data". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2971–2980. URL: `http://proceedings.mlr.press/v80/lehtinen18a.html`.

[121] Ch Leinert et al. "The 1997 reference of diffuse night sky brightness". In: *Astronomy and Astrophysics Supplement Series* 127.1 (1998), pp. 1–99.

[122] Richard M Levenson et al. "Multiplexing with multispectral imaging: from mice to microscopy". In: *ILAR Journal* 49.1 (2008), pp. 78–88.

[123] Anat Levin et al. "Image and depth from a conventional camera with a coded aperture". In: *ACM transactions on graphics (TOG)* 26.3 (2007), 70–es.

[124] Aviad Levis et al. "Gravitationally Lensed Black Hole Emission Tomography". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 19841–19850.

[125] Marc Levoy et al. "Light field microscopy". In: *ACM SIGGRAPH 2006 Papers*. 2006, pp. 924–934.

[126] Chengbo Li. "An efficient algorithm for total variation regularization with applications to the single pixel camera and compressive sensing". PhD thesis. Rice University, 2010.

[127] Shuai Li et al. "Imaging through glass diffusers using densely connected convolutional networks". In: *Optica* 5.7 (2018), pp. 803–813.

[128] Yunzhe Li, Yujia Xue, and Lei Tian. "Deep speckle correlation: a deep learning approach toward scalable imaging through scattering media". In: *Optica* 5.10 (2018), pp. 1181–1190.

[129] Orly Liba et al. "Handheld mobile photography in very low light". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–16.

[130] Xing Lin et al. "Spatial-spectral encoded compressive hyperspectral imaging". In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), pp. 1–11.

[131] Fanglin Linda Liu et al. "Fourier DiffuserScope: Single-shot 3D Fourier light field microscopy with a diffuser". In: *arXiv preprint arXiv:2006.16343* (2020).

[132] Fanglin Linda Liu et al. "Single-shot 3D fluorescence microscopy with Fourier DiffuserCam". In: *Novel Techniques in Microscopy*. Optical Society of America. 2019, NS2B–3.

[133] Jiaming Liu et al. "Image restoration using total variation regularized deep image prior". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7715–7719.

[134] Zhaoqiang Liu and Jonathan Scarlett. "Information-theoretic lower bounds for compressive sensing with generative models". In: *IEEE Journal on Selected Areas in Information Theory* (2020).

[135] Antoine Liutkus et al. "Imaging with nature: Compressive imaging using a multiply scattering medium". In: *Scientific reports* 4 (2014), p. 5552.

[136] Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. "Annotated high-throughput microscopy image sets for validation." In: *Nature methods* 9.7 (2012), pp. 637–637.

[137] Patrick Llull et al. "Coded aperture compressive temporal imaging". In: *Optics express* 21.9 (2013), pp. 10526–10545.

[138] Guolan Lu and Baowei Fei. "Medical hyperspectral imaging: a review". In: *Journal of Biomedical Optics* 19.1 (2014), p. 010901.

[139] Guolan Lu et al. "Spectral-spatial classification for noninvasive cancer detection using hyperspectral imaging". In: *Journal of Biomedical Optics* 19.10 (2014), p. 106004.

[140] Luminit. *Technical Data and Downloads*. `http://www.luminitco.com/downloads/data-sheets`. [Online; accessed 19-July-2008]. 2017.

[141] Elie Maalouf, Bruno Colicchio, and Alain Dieterlen. "Fluorescence microscopy three-dimensional depth variant point spread function interpolation using Zernike moments". In: *J. Opt. Soc. Am. A* 28.9 (Sept. 2011), pp. 1864–1870. DOI: `10.1364/JOSAA.28.001864`. URL: `http://josaa.osa.org/abstract.cfm?URI=josaa-28-9-1864`.

[142] Anant Madabhushi and George Lee. *Image analysis and machine learning in digital pathology: Challenges and opportunities*. 2016.

[143] Matteo Maggioni et al. "Nonlocal transform-domain filter for volumetric data denoising and reconstruction". In: *IEEE transactions on image processing* 22.1 (2012), pp. 119–133.

[144] Matteo Maggioni et al. "Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms". In: *IEEE Transactions on image processing* 21.9 (2012), pp. 3952–3966.

[145] Emmanuel Maggiori et al. "Convolutional neural networks for large-scale remote-sensing image classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 55.2 (2016), pp. 645–657.

[146] Kshitij Marwah et al. "Compressive light field photography using overcomplete dictionaries and optimized projections". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–12.

[147] Gary Mataev, Peyman Milanfar, and Michael Elad. "DeepRED: Deep image prior powered by RED". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019.

[148] Michael T McCann, Kyong Hwan Jin, and Michael Unser. "Convolutional neural networks for inverse problems in imaging: A review". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 85–95.

[149] James G McNally et al. "Three-dimensional imaging by deconvolution microscopy". In: *Methods* 19.3 (1999), pp. 373–385.

[150] Sofiane Mihoubi et al. "Multispectral demosaicing using pseudo-panchromatic image". In: *IEEE Transactions on Computational Imaging* 3.4 (2017), pp. 982–995.

[151] Ben Mildenhall et al. "Burst denoising with kernel prediction networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2502–2510.

[152] Ben Mildenhall et al. "Nerf in the dark: High dynamic range view synthesis from noisy raw images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16190–16199.

[153] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *European conference on computer vision*. Springer. 2020, pp. 405–421.

[154] Pavlo Molchanov et al. "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4207–4215.

[155] Kristina Monakhova et al. "Dancing under the stars: video denoising in starlight". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16241–16251.

[156] Kristina Monakhova et al. "Learned reconstructions for practical mask-based lensless imaging". In: *Optics Express* 27.20 (2019), pp. 28075–28090.

[157] Kristina Monakhova et al. *Lensless learning repository*. `https://github.com/Waller-Lab/LenslessLearning/`. Accessed: 2019-09-05. 2019.

[158] Kristina Monakhova et al. "Spectral DiffuserCam: Lensless snapshot hyperspectral imaging with a spectral filter array". In: *Optica* 7.10 (2020), pp. 1298–1307.

[159] Kristina Monakhova et al. "Untrained networks for compressive lensless photography". In: *Optics Express* 29.13 (2021), pp. 20913–20929.

[160] Vishal Monga, Yuelong Li, and Yonina C Eldar. "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing". In: *IEEE Signal Processing Magazine* 38.2 (2021), pp. 18–44.

[161] Mukesh C Motwani et al. "Survey of image denoising techniques". In: *Proceedings of GSPX*. Vol. 27. 2004, pp. 27–30.

[162] Ren Ng et al. "Light field photography with a hand-held plenoptic camera". PhD thesis. Stanford University, 2005.

[163] Thanh Nguyen et al. "Deep learning approach for Fourier ptychography microscopy". In: *Optics express* 26.20 (2018), pp. 26470–26484.

[164] Gregory Ongie et al. "Deep learning techniques for inverse problems in imaging". In: *IEEE Journal on Selected Areas in Information Theory* 1.1 (2020), pp. 39–56.

[165] Antony Orth et al. "Gigapixel multispectral microscopy". In: *Optica* 2.7 (2015), pp. 654–662.

[166] Neal Parikh, Stephen Boyd, et al. "Proximal algorithms". In: *Foundations and trends® in Optimization* 1.3 (2014), pp. 127–239.

[167] Nurmohammed Patwary and Chrysanthe Preza. "Image restoration for three-dimensional fluorescence microscopy using an orthonormal basis for efficient representation of depth-variant point-spread functions". In: *Biomed. Opt. Express* 6.10 (Oct. 2015), pp. 3826–3841. DOI: 10.1364/BOE.6.003826. URL: http://www.osapublishing.org/boe/abstract.cfm?URI=boe-6-10-3826.

[168] Yifan Peng et al. "Learned large field-of-view imaging with thin-plate optics". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), p. 219.

[169] Tobias Plotz and Stefan Roth. "Benchmarking denoising algorithms with real photographs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1586–1595.

[170] Javier Portilla et al. "Image denoising using scale mixtures of Gaussians in the wavelet domain". In: *IEEE Transactions on Image processing* 12.11 (2003), pp. 1338–1351.

[171] Mangal Prakash et al. "Fully unsupervised probabilistic noise2void". In: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE. 2020, pp. 154–158.

[172] Brandon Redding et al. "Compact spectrometer based on a disordered photonic chip". In: *Nature Photonics* 7.9 (2013), p. 746.

[173] Dikpal Reddy, Ashok Veeraraghavan, and Rama Chellappa. "P2C2: Programmable pixel compressive camera for high speed imaging". In: *CVPR 2011*. IEEE. 2011, pp. 329–336.

[174] Tal Remez et al. "Class-aware fully convolutional Gaussian and Poisson denoising". In: *IEEE Transactions on Image Processing* 27.11 (2018), pp. 5707–5722.

[175] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The little engine that could: Regularization by denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[176] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[177] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[178] Ernest Ryu et al. "Plug-and-play methods provably converge with properly trained denoisers". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5546–5557.

[179] Daniel Sage et al. "DeconvolutionLab2: An open-source software for deconvolution microscopy". In: *Methods* 115 (2017). Image Processing for Biologists, pp. 28–41. ISSN: 1046-2023. DOI: https://doi.org/10.1016/j.ymeth.2016.12.015. URL: https://www.sciencedirect.com/science/article/pii/S1046202316305096.

[180] Sujit Kumar Sahoo, Dongliang Tang, and Cuong Dang. "Single-shot multispectral imaging with a monochromatic camera". In: *Optica* 4.10 (2017), pp. 1209–1213.

[181] Vishwanath Saragadam and Aswin C Sankaranarayanan. "Programmable Spectrometry: Per-pixel Material Classification using Learned Spectral Filters". In: *2020 IEEE International Conference on Computational Photography (ICCP)*. IEEE. 2020, pp. 1–10.

[182] Pinaki Sarder and Arye Nehorai. "Deconvolution methods for 3-D fluorescence microscopy images". In: *IEEE Signal Processing Magazine* 23.3 (2006), pp. 32–45.

[183] Steve Saxe et al. "Advances in miniaturized spectral sensors". In: *Next-Generation Spectroscopic Technologies XI*. Vol. 10657. International Society for Optics and Photonics. 2018, 106570B.

[184] Uwe Schmidt and Stefan Roth. "Shrinkage fields for effective image restoration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2774–2781.

[185] Takeshi Shimano et al. "Lensless light-field imaging with Fresnel zone aperture: quasi-coherent coding". In: *Applied optics* 57.11 (2018), pp. 2841–2850.

[186] Jean-Baptiste Sibarita. "Deconvolution microscopy". In: *Microscopy Techniques* (2005), pp. 201–243.

[187] Ayan Sinha et al. "Lensless computational imaging through deep learning". In: *Optica* 4.9 (2017), pp. 1117–1125.

[188] Vincent Sitzmann et al. "End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging". In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 1–13.

[189] Vincent Sitzmann et al. "Implicit neural representations with periodic activation functions". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7462–7473.

[190] Hema Somanathan et al. "Visual ecology of Indian carpenter bees I: light intensities and flight activity". In: *Journal of Comparative Physiology A* 194.1 (2008), pp. 97–107.

[191] David G Stork and Patrick R Gill. "Optical, mathematical, and computational foundations of lensless ultra-miniature diffractive imagers and sensors". In: *International Journal on Advances in Systems and Measurements* 7.3 (2014), p. 4.

[192] Jeremias Sulam et al. "Multilayer convolutional sparse modeling: Pursuit and dictionary learning". In: *IEEE Transactions on Signal Processing* 66.15 (2018), pp. 4090–4104.

[193] Jian Sun, Huibin Li, and Zongben Xu. "Deep ADMM-Net for compressive sensing MRI". In: *Advances in neural information processing systems*. 2016, pp. 10–18.

[194] Ke Sun et al. "Deep High-Resolution Representation Learning for Human Pose Estimation". In: *CVPR*. 2019.

[195] Ke Sun et al. "Deep high-resolution representation learning for human pose estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5693–5703.

[196] Da-Wen Sun. *Hyperspectral imaging for food quality analysis and control*. Elsevier, 2010.

[197] Florent Sureau, Alexis Lechat, and J-L Starck. "Deep learning for a space-variant deconvolution in galaxy surveys". In: *Astronomy & Astrophysics* 641 (2020), A67.

[198] Jun Tanida et al. "Color imaging with an integrated compound imaging system". In: *Optics Express* 11.18 (2003), pp. 2109–2117.

[199] Jun Tanida et al. "Thin observation module by bound optics (TOMBO): concept and experimental verification". In: *Applied Optics* 40.11 (2001), pp. 1806–1813.

[200] Matias Tassano, Julie Delon, and Thomas Veit. "Fastdvdnet: Towards real-time deep video denoising without flow estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1354–1363.

[201] Andrei Nikolaevich Tikhonov. "On the solution of ill-posed problems and the method of regularization". In: *Doklady Akademii Nauk*. Vol. 151. 3. Russian Academy of Sciences. 1963, pp. 501–504.

[202] Linh Duy Tran, Son Minh Nguyen, and Masayuki Arai. "GAN-based noise model for denoising real images". In: *Proceedings of the Asian Conference on Computer Vision*. 2020.

[203] Tsung-Han Tsai and David J Brady. "Coded aperture snapshot spectral polarization imaging". In: *Applied optics* 52.10 (2013), pp. 2153–2161.

[204] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.

[205] Gregory Vaksman, Michael Elad, and Peyman Milanfar. "Patch Craft: Video Denoising by Deep Modeling and Patch Matching". In: *arXiv preprint arXiv:2103.13767* (2021).

[206] Dave Van Veen et al. "Compressed sensing with deep image prior and learned regularization". In: *arXiv preprint arXiv:1806.06438* (2018).

[207] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. "Plug-and-play priors for model based reconstruction". In: *2013 IEEE Global Conference on Signal and Information Processing*. IEEE. 2013, pp. 945–948.

[208] Richard H Vollmerhausen and Tana Maurer. "Night illumination in the visible, NIR, and SWIR spectral bands". In: *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XIV*. Vol. 5076. International Society for Optics and Photonics. 2003, pp. 60–69.

[209] Ashwin Wagadarikar et al. "Single disperser design for coded aperture snapshot spectral imaging". In: *Applied Optics* 47.10 (2008), B44–B51.

[210] Michael B Wakin et al. "An architecture for compressive imaging". In: *2006 international conference on image processing*. IEEE. 2006, pp. 1273–1276.

[211] Jingdong Wang et al. "Deep High-Resolution Representation Learning for Visual Recognition". In: *TPAMI* (2019).

[212] Wei Wang et al. "Enhancing low light videos by exploring high sensitivity camera noise". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4111–4119.

[213] Wenhai Wang et al. "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 568–578.

[214] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. "Multiscale structural similarity for image quality assessment". In: *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. Ieee. 2003, pp. 1398–1402.

[215] Zhu Wang and Zongfu Yu. "Spectral analysis based on compressive sensing in nanophotonic structures". In: *Optics Express* 22.21 (2014), pp. 25608–25614.

[216] Zhu Wang et al. "Single-shot on-chip spectral sensors based on photonic crystal slabs". In: *Nature communications* 10.1 (2019), pp. 1–6.

[217] Eric Warrant. "Vision in the dimmest habitats on earth". In: *Journal of Comparative Physiology A* 190.10 (2004), pp. 765–789.

[218] Kaixuan Wei et al. "A physics-based noise formation model for extreme low-light raw denoising". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2758–2767.

[219] John Wright and Yi Ma. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. Cambridge University Press, 2022.

[220] Yujia Xue et al. "Reliable deep-learning-based phase imaging with uncertainty quantification". In: *Optica* 6.5 (2019), pp. 618–629.

[221] Kyrollos Yanny et al. "Deep learning for fast spatially varying deconvolution". In: *Optica* 9.1 (2022), pp. 96–99.

[222] Kyrollos Yanny et al. "Miniature 3D Fluorescence Microscope Using Random Microlenses". In: *Optics and the Brain*. Optical Society of America. 2019, BT3A–4.

[223] Kyrollos Yanny et al. "Miniscope3D: optimized single-shot miniature 3D fluorescence microscopy". In: *Light: Science & Applications* 9.1 (2020), pp. 1–13.

[224] Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity". In: *IEEE Transactions on Image Processing* 21.5 (2011), pp. 2481–2499.

[225] Yuhui Yuan, Xilin Chen, and Jingdong Wang. "Object-Contextual Representations for Semantic Segmentation". In: (2020).

[226] Syed Waqas Zamir et al. "Restormer: Efficient transformer for high-resolution image restoration". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5728–5739.

[227] Chen Zhang et al. "A novel 3D multispectral vision system based on filter wheel cameras". In: *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE. 2016, pp. 267–272.

[228] Jian Zhang and Bernard Ghanem. "ISTA-Net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1828–1837.

[229] Kai Zhang et al. "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising". In: *IEEE transactions on image processing* 26.7 (2017), pp. 3142–3155.

[230] Kai Zhang et al. "Plug-and-play image restoration with deep denoiser prior". In: *arXiv preprint arXiv:2008.13751* (2020).

[231] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[232] Xuaner Zhang et al. "Zoom to learn, learn to zoom". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3762–3770.

[233] Yi Zhang et al. "Rethinking Noise Synthesis and Modeling in Raw Denoising". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 4593–4601.

[234] Yide Zhang et al. "A Poisson-Gaussian Denoising Dataset with Real Fluorescence Microscopy Images". In: *CVPR*. 2019.

[235] Ellen D Zhong et al. "CryoDRGN: reconstruction of heterogeneous cryo-EM structures using neural networks". In: *Nature methods* 18.2 (2021), pp. 176–185.

[236] Kevin C Zhou and Roarke Horstmeyer. "Diffraction tomography with a deep image prior". In: *Optics Express* 28.9 (2020), pp. 12872–12896.