Computational Image Formation: Simulators in the Deep Learning Era

Stanley H. Chan

School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907
E-mail: stanchan@purdue.edu

Abstract. *At the pinnacle of computational imaging is the co-optimization of camera and algorithm. This, however, is not the only form of computational imaging. In problems such as imaging through adverse weather, the bigger challenge is how to accurately simulate the forward degradation process so that we can synthesize data to train reconstruction models and/or integrating the forward model as part of the reconstruction algorithm. This article introduces the concept of computational image formation (CIF). Compared to the standard inverse problems where the goal is to recover the latent image x from the observation $y = \mathcal{G}(x)$, CIF shifts the focus to designing an approximate mapping $\mathcal{H}_\theta$ such that $\mathcal{H}_\theta \approx \mathcal{G}$ while giving a good image reconstruction result. The word "computational" highlights the fact that the image formation is now replaced by a numerical simulator. While matching Mother Nature remains an important goal, CIF pays even greater attention to strategically choosing an $\mathcal{H}_\theta$ so that the reconstruction performance is maximized.*

The goal of this article is to conceptualize the idea of CIF by elaborating on its meaning and implications. The first part of the article is a discussion on the four attributes of a CIF simulator: accurate enough to mimic $\mathcal{G}$, fast enough to be integrated as part of the reconstruction, provides a well-posed inverse problem when plugged into the reconstruction, and differentiable to allow backpropagation. The second part of the article is a detailed case study based on imaging through atmospheric turbulence. A plethora of simulators, old and new ones, are discussed. The third part of the article is a collection of other examples that fall into the category of CIF, including imaging through bad weather, dynamic vision sensors, and differentiable optics. Finally, thoughts about the future direction and recommendations to the community are shared. ⓒ *2023 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2023.67.6.060405]

1. INTRODUCTION

The cameras we use today are largely a variant of the pinhole camera which, according to some scientists, can be traced back to nomadic tribes of North Africa thousands of years ago. A pinhole camera is easy to understand. Many of you have probably seen Gemma Frisius's drawing [1] (See Figure 1): Light is emitted from the source, propagating through the medium, and finally arriving at a tiny pinhole. Assuming that light travels along a straight line, a scaled and inverted image is formed on the screen. Today's cameras are certainly more complicated but they arguably follow the same principle, just with slightly more advanced optical elements to bend light and a sensor to record the image intensity.

Figure 1. A schematic drawing of the pinhole camera made by Gemma Frisius in 1544 [1].

Over the past century, imaging has evolved to a much more diverse form, which we now call computational imaging [2, 3]. Unlike pinhole cameras that aim to produce a sharp image, a computational imaging system co-designs the sensing unit with the algorithm. The signal generated at the junction of sensor and algorithm may not necessarily be a sharp and clean image. It can be an unrecognizable signal, but with the magic of the algorithm, we can recover the image.

However, computational imaging should not be bounded to the co-design of a *hardware* and algorithm. In this paper, I want to the concept of *computational image formation* (CIF). CIF, in my mind, is a branch in computational imaging with the focus on selecting and optimizing the image formation process. This can be a confusing (or even redundant) idea, because the image formation process is defined by either Mother Nature or hardware. If it is Nature, e.g., shot noise due to the Poisson arrival process, we have a very precise equation describing how images are formed. If it is hardware, we have already seen many great examples such as coded aperture, coded exposure, CT, MRI, etc. So, what does CIF mean?

The core substance of CIF is a *simulator*; a differentiable, fast, and accurate simulator that can be integrated into the image *reconstruction* framework. You may wonder: "A simulator? We have a ton of simulators in physics, and they

are great. Why do we need *your* simulator?" The attributes of the simulator(s) I like to discuss here are quite different from what a physics simulator looks like. In physics, a simulator's only job is to mimic nature or processes we can observe but cannot control. In CIF, the focus of the simulator is not about matching Mother Nature unconditionally, but about maximizing the image quality of the image reconstruction algorithm. To this end, a simulator in CIF needs to have four properties:

(1) **Accurate**. It should be accurate enough to mimic the true image formation process. The level of precision is determined by the image reconstruction goal.

(2) **Fast**. It needs to be fast enough to be used to synthesize training data and/or integrated in the image reconstruction loop.

(3) **Benefits reconstruction**. It needs to improve the performance of the image reconstruction. To do so, the simulator's parameters can be updated during inference.

(4) **Differentiable**. It needs to be differentiable, so that the reconstruction neural network can be trained while having the simulator in the loop.

My goal in this paper is to conceptualize CIF and invite discussions from peers. I want to explain why CIF is important in our time. I will use atmospheric turbulence as a case study, but a few other examples will also be introduced to support my discussions. I will conclude the paper with some thoughts about moving the field forward.

2. CONCEPT OF COMPUTATIONAL IMAGE FORMATION

2.1 *Pinhole Camera*

In a pinhole camera[1], we can think of the camera being a passive device. If we use $\mathbf{x} \in \mathbb{R}^d$ to denote the ground truth image in the object plane, and $\mathbf{y} \in \mathbb{R}^m$ to denote the observed image in the image plane, the camera can be mathematically described as a mapping $\mathcal{H}$ from $\mathbf{x}$ to $\mathbf{y}$:

$$\underbrace{\mathbf{y}}_{\text{observed image}} = \underbrace{\mathcal{H}}_{\text{camera}} (\underbrace{\mathbf{x}}_{\text{true image}}). \quad (1)$$

The camera model $\mathcal{H}$ can involve lenses, color filter arrays, image sensors, etc.

A post-processing image reconstruction algorithm for a pinhole camera is to find the best estimate $\widehat{\mathbf{x}}$ by solving a certain optimization (e.g., maximum likelihood estimation or maximum-a-posteriori estimation) or running it through a neural network. For generality, the reconstruction is considered as an operator $\mathcal{R}$:

$$\widehat{\mathbf{x}} = \underbrace{\mathcal{R}(\mathbf{y}, \mathcal{H})}_{\text{reconstruction}}. \quad (2)$$

[1] A distinction is made between a conventional camera and a computational camera. As the spellings of the two appear similar, the former is referred to as a pinhole camera.
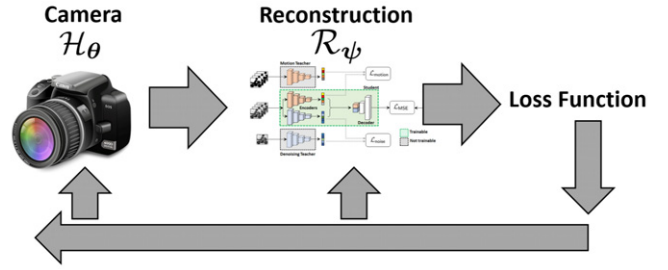

Figure 2. In a computational camera, the goal is to configure the camera parameters and the reconstruction's parameters so that the quality of the final reconstructed image is maximized.

The reconstruction method requires two inputs; the measurement $\mathbf{y}$ and the camera model $\mathcal{H}$. As a quick example, for those who do image deconvolution, $\mathcal{R}$ can be a total variation minimization:

$$\widehat{\mathbf{x}} = \mathcal{R}(\mathbf{y}, \mathcal{H}) = \underset{\mathbf{x}}{\arg\min} \|\mathcal{H}(\mathbf{x}) - \mathbf{y}\|^2 + \lambda \|\mathbf{x}\|_{\text{TV}}, \quad (3)$$

where $\mathcal{H}(\mathbf{x})$ denotes the blurring operation, and $\|\cdot\|_{\text{TV}}$ denotes the total variation norm. We can come up with many other examples which I shall skip for brevity.

The key observation of the above equations is that in a pinhole camera, we perform a capture-then-reconstruct operation. The camera is *pre-defined*. While we use the model $\mathcal{H}$ during reconstruction, we never send any *feedback* signal to $\mathcal{H}$. Furthermore, the design of $\mathcal{H}$ is separated from the reconstruction. If we ask a camera engineer to build a camera $\mathcal{H}$, he/she will never ask if we are using total variation minimization or a generative adversarial network for image reconstruction. With probability one, his/her goal is to make a $\mathcal{H}$ that produces the ideal image, for good reasons.

2.2 *Computational Camera*

In a computational camera, it is well known that the above isolation of the camera and the algorithm is replaced by a co-design philosophy. To accomplish this goal, I *parameterize* $\mathcal{H}$ with a *state vector* $\boldsymbol{\theta}$ so that the model becomes $\mathcal{H}_{\boldsymbol{\theta}}$.

The presence of the state vector θ makes the overall design interesting, as shown in Figure 2. Instead of solving a two-stage feed forward problem (design $\mathcal{H}$ and then design $\mathcal{R}$), the reconstructed image $\widehat{x}$ will be used to guide the design of $\mathcal{H}$ so that we can close the loop. By parameterizing $\mathcal{R}$ as $\mathcal{R}_{\psi}$, the end-to-end design can be written as

$$(\widehat{\psi}, \widehat{\theta}) = \underset{\psi, \boldsymbol{\theta}}{\arg\min} \; \mathbb{E}_{\mathbf{x}} \Big[\text{ReconLoss}\Big(\underbrace{\mathcal{R}_{\psi}(\mathbf{y}, \mathcal{H}_{\boldsymbol{\theta}})}_{\widehat{\mathbf{x}}}, \mathbf{x} \Big) \Big], \quad (4)$$

where ReconLoss $(\widehat{\mathbf{x}}, \mathbf{x})$ denotes the reconstruction loss between the predicted image $\widehat{\mathbf{x}}$ and the ground truth $\mathbf{x}$. I keep the definition of the loss function vague because the choice of the loss function depends on applications. It can be the squared loss, the cross-entropy loss, or any other loss that would make sense for the application.

Example 1. *(Coded aperture/lensless imaging) Consider a coded aperture camera where we are interested in recovering*

the full signal $\mathbf{x} \in \mathbb{R}^d$ from a coded measurement $\mathbf{y} \in \mathbb{R}^m$. The coding scheme we use is a fat matrix $\mathbf{H} \in \mathbb{R}^{m \times d}$ where $m \ll d$ so that the measured signal

$$\mathbf{y} = \mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x}) \overset{\text{def}}{=} \underbrace{\mathbf{H}\mathbf{x}}_{\text{encoded signal}} + \underbrace{\mathbf{n}}_{\text{noise term}}$$

has a lower dimension than the true signal $\mathbf{x}$.
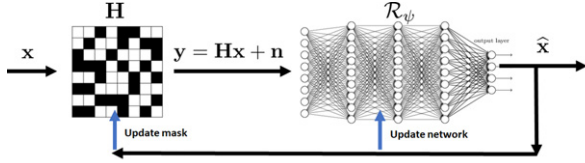
To reconstruct the signal, we create a reconstruction algorithm (e.g., a neural network) that does

$$\widehat{\mathbf{x}} = \mathcal{R}_{\psi}(\mathbf{y}).$$

Here, we can think of ψ being the weights of the neural network.

To train the neural network and simultaneously find the optimal sensing matrix $\mathbf{H}$, we perform the joint optimization

$$\widehat{\mathbf{H}}, \widehat{\psi} = \underset{\mathbf{H}, \psi}{\operatorname{argmin}} \; \mathbb{E}_{\mathbf{x},\mathbf{n}} \Big[\operatorname{ReconLoss}(\mathcal{R}_{\psi}(\mathbf{H}\mathbf{x} + \mathbf{n}), \mathbf{x}) \Big].$$

The expectation is taken with respect to both the signal $\mathbf{x}$ and noise $\mathbf{n}$ because they are random.

If we consider the hardware feasibility, we can further pose constraints on $\mathbf{H}$. For example, we can require $\mathbf{H}$ to be binary so that it can be implemented through the digital micro-mirror devices (DMD).

2.3 Nature and Simulator

The biggest difference between a computational camera and computational image formation (CIF) is the presence of a *simulator*. To explain the idea it would be useful to consider the problem of imaging through an undesirable environment, e.g., fog, haze, turbulence, or scattering medium.

Nature. The subject of CIF is often concerned with *nature*. For the time being let this be called a general degradation process, denoted by $\mathcal{G}$:

$$\underbrace{\mathbf{y}}_{\text{observed image}} = \underbrace{\mathcal{G}}_{\text{nature}} (\; \underbrace{\mathbf{x}}_{\text{true image}}). \tag{5}$$

The degradation process is unknown to us. We have no idea of how each point in the object plane is mapped to a digital value in the image plane. We can in theory run procedures such as ray tracing to find out how each ray is distorted. But there will be infinite rays to trace and so realistically we can never perform such calculations. In some situations such as random scattering medium, we cannot even trace rays because it is impossible to know how each molecule affects the light.

Example 2. *(Modeling haze) One of the landmark papers in imaging through weather is the dark channel prior by He et al. published in CVPR 2009 [4], although the modeling can be*

traced back to Fattal [5], Rossum and Nieuwenhuizen [6] and Koschmieder [7].

When light propagates through a scattering medium, the water molecules along the propagation path will cause attenuation of the light. The exact image formation process is governed by nature, and is unknown. We denote it as $\mathcal{G}$.

To simulate the image formation, people use the so-called radiative transport equation by modeling the overall degradation as a sum of the airlight term and the direct attenuation term. This gives us a simulator $\mathcal{H}_{\boldsymbol{\theta}}$:

$$\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x} \odot \mathbf{t} + \mathbf{A}(\mathbf{1} - \mathbf{t}),$$

where the state vector θ contains the airlight color $\mathbf{A}$ and the transmission map $\mathbf{t}$, with $\odot$ defining the elementwise multiplication. Note that $\mathcal{H}_{\boldsymbol{\theta}}$ is a very good model for $\mathcal{G}$, but $\mathcal{H}_{\boldsymbol{\theta}} \neq \mathcal{G}$.
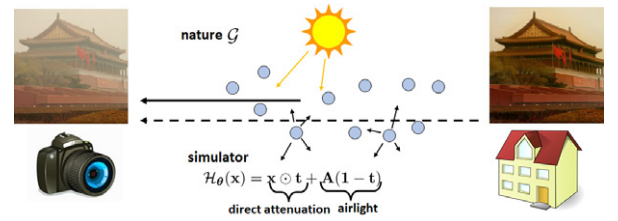
Simulator. Since $\mathcal{G}$ is unknown, we need to approximate it using a model $\mathcal{H}_{\boldsymbol{\theta}}$. I call it a simulator. A simulator, by construction, reproduces *part* of the nature. Therefore, the output of the simulator is not $\mathbf{y}$ but

$$\widehat{\mathbf{y}} = \mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x}). \tag{6}$$

I emphasize that $\mathcal{H}_{\boldsymbol{\theta}}$ is a *parameterization* of nature. The parameterization is a dimensionality reduction of the operator $\mathcal{G}$ from an infinite-dimensional space to a finite (and often low) dimensional space. The choice of the parametric model is often based on physics, but there are also man-made parameterizations. The following is an example.

Example 3. *(Environment parameterization) In Ref. [8], Gnanasambandam et al. proposed the idea of optically perturbing the appearance of an object by using structured illumination. The concept was that given a ground truth image $\mathbf{x}$ associated with a class label ℓ, an optical system $\mathcal{H}_{\boldsymbol{\theta}}$ can be built such that the distorted image $\widehat{\mathbf{y}} = \mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x})$ will be misclassified as label ℓ'. The optical system was implemented using a projector-camera setup, where the projector illuminates the real 3D object using a calculated pattern. This provides a mechanism to test the resilience of the classifier to manipulated attacks.*
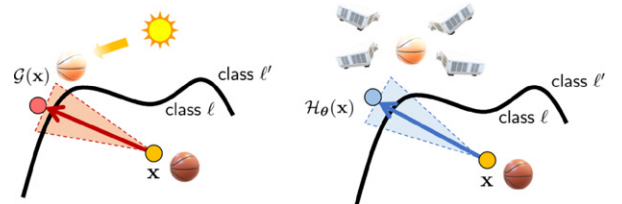
What is relevant to CIF is that the same optical setup can be used to analyze the robustness against natural
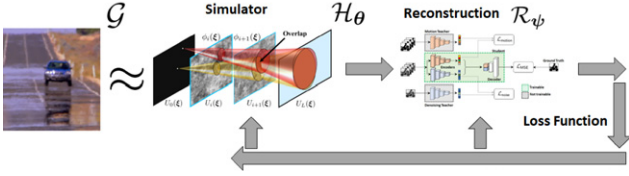
Figure 3. In Computational Image Formation (CIF), the true image formation process is determined by nature $\mathcal{G}$. The design task here is to select and optimize a simulator $\mathcal{H}_{\boldsymbol{\theta}}$ such that it not only achieves a desirable modeling accuracy, but more importantly maximizes the reconstruction quality.

perturbations, such as shadow or overcast. These natural perturbations are $\mathcal{G}$, of which the exact forms are unknown to us. Some people use graphics rendering (e.g., for flight simulator) to synthesize the scenes. With the projector-camera system, we can use the multiple projectors to parameterize nature by using a finite number of tunable knobs to reproduce outdoor environment in a controlled setting, hence providing an approximation $\mathcal{H}_{\boldsymbol{\theta}} \approx \mathcal{G}$.

The presence of nature and a simulator changes the problem scope from designing a hardware camera to designing a numerical simulator as shown in Figure 3. Because of its middle-person role between nature and algorithm, the selection and optimization of the simulator is significantly more complicated than configuring a camera such as deciding which coded aperture pattern to use. An analogy here is that selecting a simulator is like selecting a neural network, whereas configuring a camera is like learning the weight of the neural network.

2.4 *Four Attributes of a CIF Simulator*
In what follows, I will describe the four attributes of the simulator in CIF.

Attribute 1: Accurate. The first attribute of a CIF simulator $\mathcal{H}_{\boldsymbol{\theta}}$ is that it is accurate enough to approximate the true image formation process $\mathcal{G}$. The form of $\mathcal{H}_{\boldsymbol{\theta}}$ can be anything. It can be a one-line equation, an algorithmic procedure, or even a neural network. The performance of $\mathcal{H}_{\boldsymbol{\theta}}$ is measured in terms of how good the simulated image is when compared to the true image. Mathematically, we define the loss as

$$\mathcal{E}_{\text{sim}}(\mathcal{H}_{\boldsymbol{\theta}}) \overset{\text{def}}{=} \underbrace{\mathbb{E}_{\mathbf{x}}\Big[\text{SimLoss}(\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x}), \mathcal{G}(\mathbf{x}))\Big]}_{\text{simulator loss}}. \tag{7}$$

The equation says that for a given image $\mathbf{x}$, the simulated image $\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x})$ needs to be sufficiently similar to $\mathcal{G}(\mathbf{x})$. The goodness of $\mathcal{H}_{\boldsymbol{\theta}}$ is measured according to the loss function SimLoss. For example, in atmospheric turbulence, SimLoss can be the pixelwise distance between the simulated long/short exposure function and the measured (real) long/short exposure function. It can also be the differential tilt statistic or the z-tilt statistic between the simulation and the theoretically derived statistical average. Or, SimLoss can be measured using target patterns seen through a heat chamber. There are also instrumentations such as the

scintillometer to measure the turbulence profile. In any case, SimLoss is an application specific metric(s).

Attribute 2: Fast. The second attribute of a CIF simulator is that it needs to be fast so that we can integrate it as a part of the reconstruction algorithm. But speed depends on which computing platform we use. (See discussions in the next paragraph.) A slightly better way is to measure the complexity of the model, with the following notation

$$\mathcal{E}_{\text{complexity}}(\mathcal{H}_{\boldsymbol{\theta}}) \tag{8}$$

The definition of the complexity can take various forms. If $\mathcal{H}_{\boldsymbol{\theta}}$ is implemented using a deep neural network, then the complexity can be measured in terms of the number of hidden layers, width of hidden layers, number of parameters, number of filters, size of the filters, etc. The complexity of $\mathcal{H}_{\boldsymbol{\theta}}$ can also be viewed at two levels: (1) The intrinsic capacity of the model and (2) The effective model capacity given a specific choice of θ.

Model complexity is usually linked to speed (aka runtime), but some caveats should be noted. For example, 2D convolutions today can be efficiently implemented because of the dedicated hardware architectures on graphics processing units. In contrast, depth-wise 3D convolutions are substantially slower even if the number of parameters is the same as a 2D convolution because of the lack of specialized hardware. While this problem will likely be solved in the near future, the complexity is not translated to runtime directly.

Attribute 3: Benefits Reconstruction. The third attribute of a CIF simulator is that it needs to benefit the reconstruction performance because in CIF, the ultimate goal is to maximize the final image produced by the entire system.

The reconstruction loss is abstractly defined as

$$\mathcal{E}_{\text{recon}}(\mathcal{H}_{\boldsymbol{\theta}}) \overset{\text{def}}{=} \min_{\mathcal{R}_{\psi}} \underbrace{\mathbb{E}_{\mathbf{x}}\Big[\text{ReconLoss}\Big(\mathcal{R}_{\psi}(\mathbf{y}, \mathcal{H}_{\boldsymbol{\theta}}), \mathbf{x}\Big)\Big]}_{\text{reconstruction loss}}. \tag{9}$$

The reconstruction loss in Eq. (9) is reminiscent to Eq. (4), but the goals are very different. In Eq. (4), the camera $\mathcal{H}_{\boldsymbol{\theta}}$ is already chosen except for its parameter θ. Thus, the joint minimization in Eq. (4) is to simultaneously optimize for the reconstruction (realized via a neural network) and the state vector θ of the camera. In contrast, the minimization problem in Eq. (9) says that we do not even know which simulator to use. Using atmospheric turbulence as an example (see Section 3), the decision could be the choice between two completely different models, e.g., the pixel-jitter model or the deformable model. Therefore, in CIF simulator selection, the difficulty is how to *build* a simulator that fulfills the criteria instead of adjusting parameters of some simple equations.

Attribute 4: Differentiable. The fourth attribute of a simulator is that it needs to be differentiable. The metric for differentiability can be a simple indicator function:

$$\mathcal{E}_{\text{diff}}(\mathcal{H}_{\boldsymbol{\theta}}) \overset{\text{def}}{=} \begin{cases} 0, & \text{if } \mathcal{H}_{\boldsymbol{\theta}} \text{ is differentiable}, \\ \infty, & \text{if } \mathcal{H}_{\boldsymbol{\theta}} \text{ is not differentiable}. \end{cases} \tag{10}$$

Why do we need differentiability? The reason is that most, if not all, image reconstruction algorithms today are based on deep neural networks. For the simulator to be part of the reconstruction framework, it is necessary that the gradient of the loss function can be backpropagated to the input. Therefore, being able to take the gradient of the simulator with respect to its model parameters becomes essential.

Enforcing differentiability can be realized in multiple ways. The easiest way, of course, is to build a simulator using a deep neural network. However, speaking from experience, physics-based simulators often offer significantly better guarantees of the theoretical properties and are more interpretable. Yet, physics-based simulators are often complicated. Enforcing differentiability is not a trivial task. For example, iterative procedures such as Newton's method, which is widely used to find the intersection of a ray and the lens surface, needs to be replaced by another method that is non-iterative before we can make it differentiable.

2.5 *Simulator Selection and Parameter Optimization*
After stating the four attributes of the simulator, I would like to elaborate on the simulator selection problem and the parameter optimization problem. The two problems are different. The former is more about *building* a simulator that fulfills the criteria. The latter assumes that the candidate simulator has already been decided, but during the inference process, we would like to update the state vector to better match with the observation. The simple analogy here is that simulator selection picks the number of partitions in a spatially varying blur system (i.e., how many pixels to share one blur kernel), whereas the parameter update assumes that the partition is already fixed but we need to estimate the individual blur kernels.

Simulator Selection. The simulator selection problem can be formulated as a constrained optimization.

$$\begin{aligned}
\underset{\mathcal{H}_{\boldsymbol{\theta}}}{\arg\min} \quad & \mathcal{E}_{\text{recon}}(\mathcal{H}_{\boldsymbol{\theta}}) + \mathcal{E}_{\text{diff}}(\mathcal{H}_{\boldsymbol{\theta}}) \\
\text{subject to} \quad & \mathcal{E}_{\text{complexity}}(\mathcal{H}_{\boldsymbol{\theta}}) \leq \tau_{\text{complexity}}, \\
& \mathcal{E}_{\text{sim}}(\mathcal{H}_{\boldsymbol{\theta}}) \leq \tau_{\text{sim}}
\end{aligned} \tag{11}$$

The objective function of the problem is the reconstruction loss $\mathcal{E}_{\text{recon}}(\mathcal{H}_{\boldsymbol{\theta}})$ plus the indicator function $\mathcal{E}_{\text{diff}}(\mathcal{H}_{\boldsymbol{\theta}})$. Since by construction the simulator needs to be differentiable, the indicator function will serve the purpose of only allowing differentiable simulators to be considered. The main objective is therefore the reconstruction loss.

There are two constraints in the optimization. The complexity constraint limits how complex the simulator can be. The simulator loss constraint controls the match between the simulator $\mathcal{H}_{\boldsymbol{\theta}}$ and the ground truth model $\mathcal{G}$. The reason why these two criteria are listed as constraints instead of the objective is that in practice, the simulator designer would seldom navigate the four criteria simultaneously when choosing the simulator. It is far more common to develop a simulator with a certain level of complexity and simulation

accuracy in mind, while leaving some design freedom to the reconstruction algorithm.

I should stress that while the simulator selection problem appears like an ordinary constrained optimization, the problem is never solved using any gradient based methods. In contrast, the problem is more often solved manually based on experience and creativity. So it is more of an art.

Parameter Optimization. To stimulate the parameter update discussion, it is best to first consider an example.

Example 4. *(Blind deconvolution) In the classical blind image deconvolution problem, we are often interested in the alternating minimization*

$$\mathbf{h}^{k+1} = \underset{\mathbf{h}}{\arg\min} \|\mathbf{h} \circledast \mathbf{x}^k - \mathbf{y}\|^2 + \text{prior}(\mathbf{h}),$$

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}}{\arg\min} \|\mathbf{h}^{k+1} \circledast \mathbf{x} - \mathbf{y}\|^2 + \text{prior}(\mathbf{x}),$$

where prior$(\cdot)$ is a generic notation representation the prior model of the respective variable.

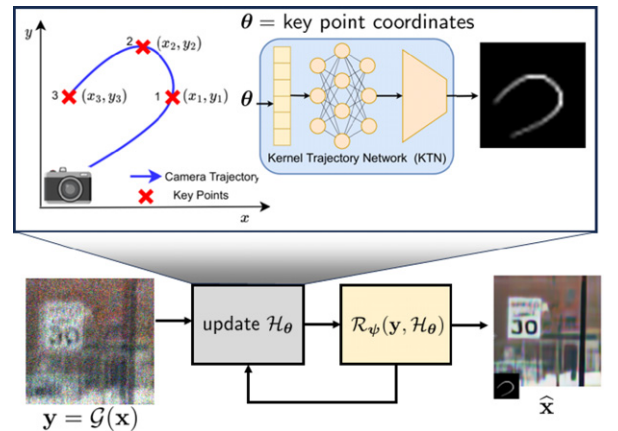*The first equation, put into the CIF framework, can be rewritten as*

$$\theta^{k+1} = \underset{\theta}{\arg\min} \left\{ \text{SimLoss}\left(\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x}^k), \mathcal{G}(\mathbf{x})\right) \right\}.$$

Here, $\mathcal{G}$ is the true image formation process given by nature, and $\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{h} \circledast \mathbf{x} + \mathbf{n}$ is the approximation. The second equation can be written as

$$\mathbf{x}^{k+1} = \mathcal{R}_{\psi}(\mathbf{y}, \mathcal{H}_{\theta^k}),$$

which is a generic form of the image reconstruction method.

To be precise about the state vector θ, one can think of it as the parametric description of the forward model $\mathcal{H}_{\boldsymbol{\theta}}$. For example, in Ref. [9], key points are used to characterize the motion where a continuous curve is drawn to connect the key points to generate a motion blur kernel. The key points are coordinates in the 2D space, and the state vector θ is the collection of all key points.

The above example illustrates the typical procedure of updating the simulator's parameters and estimating the latent image. These two steps can be summarized by the following

Table I. Comparison between a physics simulator and CIF simulator.

	Physics simulator	CIF simulator
Goal	Describe nature	Help image reconstruction
Accurate?	As much as possible because we need it to reproduce nature.	No need to be absolutely accurate. As long as it meets a certain level, it is okay.
Fast?	Fast is certainly welcome, but slow is also okay if it is justified by application.	Need to be very fast, for synthesizing datasets and integrating with inverse algorithms.
Benefits Reconstruction?	Irrelevant because the simulator's goal is to reproduce nature.	Simulator in the reconstruction loop is the key of CIF.
Differentiable?	Irrelevant. No need	Needed, especially if we want to co-optimize reconstruction and simulator.

pair of iterative updates:

$$\mathbf{x}^{k+1} = \mathcal{R}_\psi(\mathbf{y}, \ \mathcal{H}_{\theta^k}) \tag{12}$$

$$\theta^{k+1} = \underset{\theta}{\arg\min} \ \text{SimLoss}(\mathcal{H}_\theta(\mathbf{x}^{k+1}), \mathcal{G}(\mathbf{x})) \tag{13}$$

As mentioned earlier, this pair of equations are performed *after* the simulator is selected.

2.6 Discussions

Ideal $\mathcal{G}$? At this point, it should be clear why a perfect simulator $\mathcal{H}_\theta = \mathcal{G}$ may not be a good simulator in CIF. If $\mathcal{G}$ is intrinsically complicated, then matching it would require a complex $\mathcal{H}_\theta$. This will violate Attribute 2 (simple and fast), and possibly Attribute 4 (differentiable). There is also a possibility that Attribute 3 (improve reconstruction) can be poor because a complex $\mathcal{H}_\theta$ often leads to an ill-posed inverse problem. The reconstruction model $\mathcal{R}_\psi$ has to be even more complex in order to invert the effect of $\mathcal{G}$. This means that we have a good forward model, but we cannot solve the inverse problem.

Compared to a physics simulator. Compared to a pure physics simulator, a CIF simulator has additional emphasis on speed, reconstruction, differentiability and compatibility with the reconstruction algorithm. This leads to the summary in Table I.

Compared to computational camera. How is CIF different from a computational camera? In a computational camera, the optimization variable is the hardware camera. For example, we adjust the lens parameter, or the mask in coded aperture, or the pattern of the exposure multiplexing scheme. CIF *can*, in principle, be a superset because these hardware elements can always be described by mathematical models. However, the bigger difference (and also the problem context) is that CIF is more relevant to situations where $\mathcal{G}$ is not easily parameterized by simple equations. Weather is the best example to think about. In those cases, *choosing* which $\mathcal{H}_\theta$ is more relevant than optimizing for θ. In short, CIF is not only about the co-optimization of the image formation elements, but also about the selection of the image formation model.

Compared to model selection literature. The subject of simulator selection is in many ways similar to the classical topic on model selection [10]. The core difference, however,
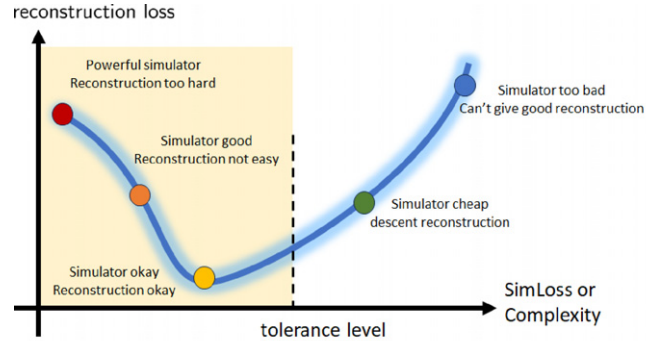

Figure 4. A conceptual illustration of the trade-off between the performance of the simulator and the image reconstruction.

is that in model selection, the models are more or less simple equations; for example, the ℓ_1-norm (LASSO) or the ℓ_2-norm (ridge). In CIF, the selection of simulator can be drastically more difficult. Often times, there may not be any readily available simulators to "choose from". The designer needs to develop a simulator to match the complexity and simulation loss, while maximizing the image reconstruction performance. This is *not* a simple gradient search problem, but more of a design problem.

Because of the different goals and different problem settings, the methodologies used in model selection largely do not apply. For example, tools such as the Akaike information criterion (AIC) or the Bayesian information criterion (BIC) are not applicable because (i) even though there is an underlying probability distribution of the true data, we do not know what it is, (ii) most simulators are complex numerical procedures. Analyzing them and calculating in the context of AIC and BIC is not feasible, (iii) there exist domain-specific alternative assessment techniques that are much more reliable and direct than AIC and BIC.

Trade-off. Because of the conflicting goals in the simulator selection problem, it would be useful to see the tension via the illustration in Figure 4. In this figure, the y-axis defines the reconstruction loss whereas the x-axis defines the simulator loss (or complexity). The question of CIF is about which point on the trade-off curve to pick.

On the far right of the figure, the simulator is too simple to be of any use. Even though the reconstruction is easy, the reconstructed image will not be good. Moving slightly to the left, the simulator is improved and the reconstruction is also improved, thanks to a more meaningful simulator. The benefit can come from the data synthesis (i.e. to generate a good training dataset), as well as in-the-loop inverse algorithm. The optimal operating point is somewhere in the middle, where the simulator is reasonably accurate and the reconstruction is also decent. As the simulator moves towards the left hand side, the reconstruction becomes more difficult. Not only do we have a harder time to synthesize enough data because the simulation can be time-consuming, but we also have a harder time to put the simulator in the reconstruction loop. As the simulator moves to the far left, the simulator is extremely good but it offers little to no help to the reconstruction. Thus, the performance can be very bad. In Fig. 4, a colored region is marked with a cutoff τ. Any simulator that gives a performance better than τ will be inside the region. Therefore, among all the possible simulators, only those that meet the simulator criteria can be adopted for reconstruction.

3. CASE STUDY: IMAGING THROUGH TURBULENCE

Now that we have seen what CIF means, it is time to look at a concrete example. I will use turbulence as an illustration because of the richness of the literature. To give you an idea of the accuracy of the model, I will use the number of stars ★ is used to indicate how accurate the simulator is. A simulator with one (★) means that it is cheap but inaccurate, whereas a simulator with (★★★★★) means that it is extremely accurate. The number of stars is partially objective because we can measure the quality, but it is also partially subjective because much of these are based on my experience.

3.1 *Wave Propagation Physics*

Our atmosphere is a complicated medium because of its turbulent nature. Factors such as temperature, wind velocity, humidity, and other weather conditions can affect the strength of turbulence. As light propagates, the random index of refraction will cause phase delays along the optical paths, leading to pixel displacements and blurs [11, 12]. The exact image formation for one specific turbulent instance cannot be determined because it is stochastic.

Split-step ("Gold standard") (★★★★★) Let me start by discussing the most accurate model so that we have a reference point. In physics, the most accurate model for atmospheric turbulence, to date, is the split-step propagation [13]. The model says that we can partition the propagation path into a sequence of segments where each segment has a random phase screen determining the phase distortion. The propagation is performed by integrating the electromagnetic field $\mathbf{u}$ of which the magnitude produces the image $\mathbf{x} = |\mathbf{u}|^2$.

Without diving into the technical details, we can think of the split-step propagation as a sequence of operations
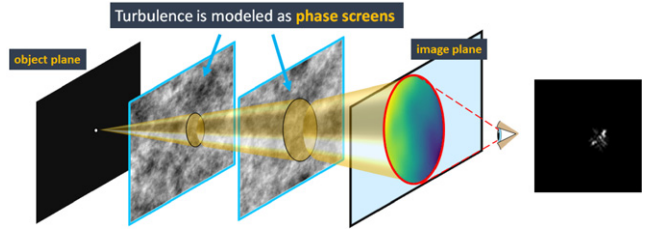

Figure 5. The split-step propagation model $\mathcal{H}_{\boldsymbol{\theta}}^{\text{split}}$ is considered as the "gold standard" in modeling atmospheric turbulence. While the model is accurate, its numerical wave propagation equation makes the implementation extremely difficult and very slow. Thus, it is a good simulator for mimicking nature, but not so much for image restoration.

along the light propagation path. At the ith segment, the electromagnetic field propagates according to the equation

$$\mathbf{u}_{i+1} = \text{Fresnel} (\ \underbrace{\mathbf{u}_i}_{i\text{th field}} \ \odot \ \underbrace{e^{j\phi_i}}_{i\text{th phase screen}}), \qquad (14)$$

where $\odot$ denotes elementwise multiplication, Fresnel($\cdot$) denotes the Fresnel diffraction integral [14, 15], and ϕ_i denotes the ith phase screen drawn from the Kolmogorov power spectral density [16–18]. (Assuming a "typical" situation with a small to moderate Rytov number so that we can ignore the amplitude effect and the power spectral density roughly follows the 5/3-power law predicted by Kolmogorov [19].) The process is nonlinear because for every pixel in the object plane, the equation has to be evaluated repeatedly (which includes Fourier transform, phase cropping, and multiplication), as illustrated in Figure 5. For completeness, the equation is written as

$$\widehat{\mathbf{y}} = \underbrace{|\text{Fresnel}(\text{Kolmgrv}(\ldots(\text{Fresnel}(\text{Kolmgrv}(\mathbf{x}))))) |^2}_{\mathcal{H}_{\boldsymbol{\theta}}^{\text{split}}(\mathbf{x})},$$

$$(15)$$

where $\mathcal{H}_{\boldsymbol{\theta}}^{\text{split}}$ denotes the split-step model, with θ defining the random phase screens along the propagation path.

For decades, physicists have agreed that the split-step propagation is very reliable and accurate for simulating atmospheric turbulence. With a sufficient number of phase screens along the path, and assuming that the turbulence is at the most moderately strong, it is safe to say that

$$\mathcal{G}(\mathbf{x}) \approx \mathcal{H}_{\boldsymbol{\theta}}^{\text{split}}(\mathbf{x}). \qquad (16)$$

Various simulation packages have been built [20–30], and they are widely used to study the optical communication, high energy beam propagation, and astronomical applications. There are also attempts to use these simulators for image processing tasks, but scholars generally find split-step too slow for any real-time applications. The bigger problem arises if we want to use $\mathcal{H}_{\boldsymbol{\theta}}^{\text{split}}$ for image restoration. Let's see why.

3.2 *Image Restoration*

Split-step propagation is never used as part of the image restoration pipeline. I will illustrate this using two reconstruction methods.

The first approach is the optimization approach. Using total variation as one of the (many) examples, we have

$$\widehat{x} = \mathcal{R}_\psi(\mathbf{y}) = \underset{\mathbf{x}}{\operatorname{argmin}} \; \|\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{split}}(\mathbf{x}) - \mathbf{y}\|^2 + \lambda\|\mathbf{x}\|_{\mathrm{TV}}. \quad (17)$$

But solving this problem can be difficult because $\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{split}}$ is complicated. If we perform variable-splitting techniques such as the ADMM algorithm, we have no way of computing the proximal map because the inverse (or regularized inverse) $(\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{split}})^{-1}$ is not possible to obtain. Note that I am not even talking about estimating the state vector θ, which will add more difficulties to the problem.

Another approach is deep learning. The idea is to train a deep neural network $\mathcal{R}_\psi$ that performs

$$\widehat{\mathbf{x}} = \mathcal{R}_\psi(\mathbf{y}, \mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{split}}). \quad (18)$$

However, the problem is in generating the clean-noisy pair for training. We can send the clean image through $\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{split}}$, and repeat the process until we have created a dataset of training samples. This is in theory doable, but since the simulator is so slow, there is no way we can synthesize enough data [31].

There is one more issue about deep learning. If we want to incorporate $\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{split}}$ into the reconstruction neural network $\mathcal{R}_\psi$, the simulator needs to be *differentiable*. Since the split-step simulation is a serial chain of complex operations going back and forth with the Fourier transforms per pixel, back propagating the gradient of the loss function through the simulator is nearly impossible unless we introduce additional approximations.

CIF Trade-off for Split-step.

- $\mathcal{E}_{\mathrm{sim}}$: Extremely small, especially if we use enough phase screens.
- $\mathcal{E}_{\mathrm{complexity}}$: Huge.
- $\mathcal{E}_{\mathrm{recon}}$: Unknown, because the only way split-step can be used is to synthesize training data. But even so, it is computationally infeasible because split-step is too slow.
- $\mathcal{E}_{\mathrm{diff}}$: Infinity, because split-step needs to perform a sequential chain of Fourier transforms of phase functions per pixel.

3.3 *Lightweight Simulators*

Having discussed the "gold standard" split-step simulator, let us move on to something faster.

Pixel Jitter Model [20]. (⋆) The simplest simulator reported in the literature is the pixel shift + blur model. This model says that

$$\widehat{\mathbf{y}} = \underbrace{\mathbf{h} \circledast (\mathrm{jitter}(\mathbf{x}))}_{\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{jitter}}(\mathbf{x})}, \quad (19)$$

where jitter($\cdot$) denotes the pixel jittering, where the displacement is a random vector with an independent and identically distribution (i.i.d). The operator $\mathbf{h}$ is a spatially-invariant Gaussian blur, and $\circledast$ is the convolution. Figure 6 shows an example. $\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{jitter}}$ is very cheap and fast – just draw a random i.i.d. motion vector field (pick any distribution, e.g., Gaussian), and convolve the image with an invariant blur kernel. The state vector θ contains the blur and the motion vector field.

Needless to say, inverting $\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{jitter}}$ is easy, at least in the non-blind case where we know the random vector field and the blur; just remove the blur using any off-the-shelf non-blind deconvolution algorithm, and then un-do the pixel displacement. We can plug it into a neural network. We can also plug it into a classical optimization so we can solve the proximal map. The only caveat (and a big caveat) is that $\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{jitter}} \not\approx \mathcal{G}$, not even close.

CIF Trade-off for Pixel-Jitter Model.

- $\mathcal{E}_{\mathrm{sim}}$: Huge, because the simulator is too simple.
- $\mathcal{E}_{\mathrm{complexity}}$: Extremely small.
- $\mathcal{E}_{\mathrm{recon}}$: Poor, because the simulator is too far from reality although solving the reconstruction problem is easy. Data synthesized by the model will have a hard time generalizing to real turbulence.
- $\mathcal{E}_{\mathrm{diff}}$: Zero, because both pixel jitter and blurring steps are differentiable.

Deformable Model [32–36]. (⋆⋆) A slightly better version of the simulator is the deformable grid model. Instead of assuming an i.i.d. random displacement field, we select a small set of anchor points in the image grid and perform a non-rigid deformation. The overall equation is

$$\widehat{\mathbf{y}} = \underbrace{\mathbf{h} \circledast (\mathrm{deform}(\mathbf{x}))}_{\mathcal{H}_{\boldsymbol{\theta}}^{\mathrm{deform}}(\mathbf{x})}, \quad (20)$$

The benefit of the deformable model is that it is physically more meaningful because the pixel displacement caused by turbulence is spatially correlated. The deformable model provides a way to enforce the spatial correlation. Figure 7 illustrates an example, where we can see that the deformable grid is more structured. The sampling process is reasonably easy because we only need to identify a few anchor points.

The deformable grid facilitates the estimation of the pixel displacement because the number of anchor points is finite. The drawback, though, is that the blur remains spatially invariant. It is still valid for isoplanatic turbulence, but definitely not true for anisoplanatic turbulence where the blur must be spatially varying.
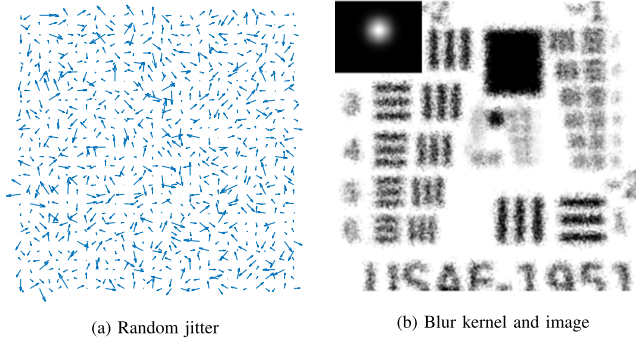
(a) Random jitter   (b) Blur kernel and image

Figure 6. Random jitter and spatially invariant Gaussian blur. The simulated turbulence effect is quite far from the reality.

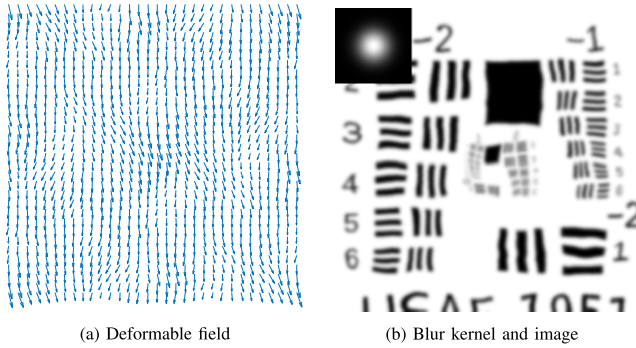(a) Deformable field   (b) Blur kernel and image

Figure 7. Deformable field and spatially invariant Gaussian blur. The simulated turbulence effect is better than the random jitter, but still not accurate enough to match the real turbulence.

CIF Trade-off for Deformable Model.

- $\mathcal{E}_{\text{sim}}$: Better than the pixel-jitter model.
- $\mathcal{E}_{\text{complexity}}$: Small. Comparable to pixel-jitter.
- $\mathcal{E}_{\text{recon}}$: Not great but better than pixel-jitter. The simulator is still far from reality although solving the reconstruction problem is easy. Data synthesized by the model does not generalize to real turbulence.
- $\mathcal{E}_{\text{diff}}$: Zero. The model is differentiable.

Varying Blur Model. (⋆⋆⋆ Incomplete) To model anisoplanatic turbulence, people propose to adopt a spatially varying blur. Keeping the deformable grid idea (which can be replaced by other pixel displacement models), the output image is defined in a per-pixel basis:

$$\widehat{\mathbf{y}}_i = \underbrace{\mathbf{h}_i \circledast (\text{deform}(\mathbf{x}))}_{\mathcal{H}_{\boldsymbol{\theta}}^{\text{varying}}(\mathbf{x})}, \qquad (21)$$

where i denotes the ith pixel of the image $\widehat{\mathbf{y}}$ and $\mathbf{h}_i$ denotes the blur for the ith pixel. Because of the per-pixel blur, this model can now describe a wider set of turbulence effects. However, the generation of these per-pixel blurs remains unclear (that's
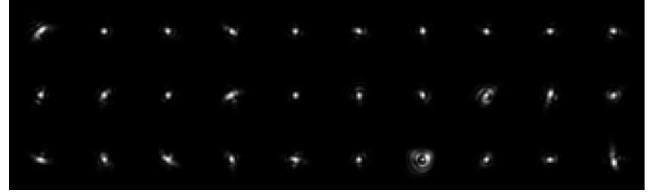

Figure 8. Spatially varying blur kernels used in $\mathcal{H}_{\boldsymbol{\theta}}^{\text{varying}}$. The kernels are generated using the phase-over-aperture model. Note that the shape of the blur changes from one location to another.

why I mark it as "incomplete"). In the example blur kernels I show in Figure 8, I use the phase-over-aperture model (to be discussed in the next subsection) to generate the spatially varying blur. These blurs are then integrated into this model $\mathcal{H}_{\boldsymbol{\theta}}^{\text{varying}}$ to perform the distortion.

The inverse problem associated with $\mathcal{H}_{\boldsymbol{\theta}}^{\text{varying}}$ is not easy, because spatially varying blurs do not have simple inverses using standard tools such as the Fourier transform. Therefore, additional approximations are needed to reduce complexity.

CIF Trade-off for Varying Blur Model.

- $\mathcal{E}_{\text{sim}}$: Undetermined, because it is unclear how the per-pixel blur kernels are defined while satisfying the spatial correlation of the blur.
- $\mathcal{E}_{\text{complexity}}$: Undetermined, because it depends on how the kernels are generated. Assuming that the covariance matrix is given, then sampling from it is not too complicated.
- $\mathcal{E}_{\text{recon}}$: Small if the spatial correlation of the blur kernels are satisfied.
- $\mathcal{E}_{\text{diff}}$: Zero. The model is differentiable if the blur kernels are predetermined and stored.

Flipped Model [35, 37, 38]. (⋆⋆⋆) Because of the difficulty of inverting the spatially varying blur, a work-around solution has been proposed by observing that turbulence-distorted images exhibit a lucky effect. From time to time, and from pixel to pixel, there are instants where the turbulence distortion is weak. The lucky pixels / patches existing in the raw video input are blur-free and displacement-free. Therefore, if we can identify these lucky patches, then we can mitigate majority of the turbulence effects, leaving only the diffraction limited blur. This two step procedure can be summarized as

$$\widehat{\mathbf{x}} = \underbrace{\text{deblur}\left(\text{lucky}(\mathbf{y})\right)}_{\text{reconstruction}}. \qquad (22)$$

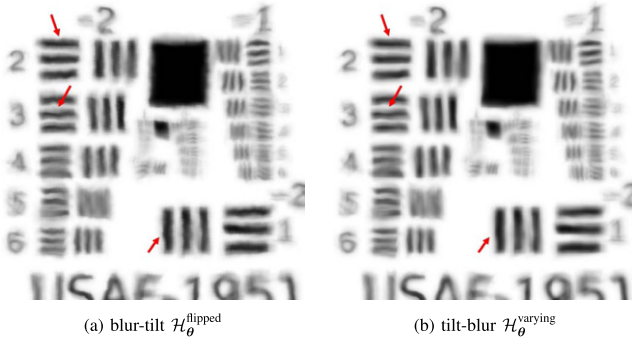There are many variations of this approach, such as [32, 36, 39–41].

(a) blur-tilt $\mathcal{H}_{\boldsymbol{\theta}}^{\text{flipped}}$ (b) tilt-blur $\mathcal{H}_{\boldsymbol{\theta}}^{\text{varying}}$

Figure 9. Two models: Tilt-then-blur $\mathcal{H}_{\boldsymbol{\theta}}^{\text{varying}}$ and blur-then-tilt $\mathcal{H}_{\boldsymbol{\theta}}^{\text{flipped}}$. They generate similar result for most regions but not for edge pixels. As shown in [42], $\mathcal{H}_{\boldsymbol{\theta}}^{\text{flipped}}$ has a tendency to "destroy" the blurs.

The two-step procedure inspired researchers to consider an alternative model where we flip the order of blur and pixel displacement in the simulator. This gives us the flipped model:

$$\widehat{\mathbf{y}}_i = \underbrace{\text{deform}(\mathbf{h}_i \circledast \mathbf{x})}_{\mathcal{H}_{\boldsymbol{\theta}}^{\text{flipped}}(\mathbf{x})}, \tag{23}$$

which according to [42], is called the blur-tilt model. The blur-tilt model is not the same as the tilt-blur model because it can be proved that wave propagation follows tilt-blur but not blur-tilt. However, the blur-tilt model is easier from the angle of inverse problem because inverting the deformation can be realized by lucky imaging algorithms.

CIF Trade-off for Flipped Model.

- $\mathcal{E}_{\text{sim}}$: Worse than the varying model because we can theoretically prove that the order of blur and tilt should be tilt-then-blur.
- $\mathcal{E}_{\text{complexity}}$: Same complexity as the varying model because it is the same composition of blur and tilt.
- $\mathcal{E}_{\text{recon}}$: Comparable to the varying blur model but the flipped model is easier to implement and hence it facilitates image reconstruction.
- $\mathcal{E}_{\text{diff}}$: Zero. The model is differentiable.

Summary. As you can see, the consideration of which $\mathcal{H}_{\boldsymbol{\theta}}$ to use is not a simple question of matching physics. If it was, then any of the cheap models would have no value. But the fact that they have been used in the literature and sometimes generated satisfactory results speak of their validity. They are not justified from the angle of physics, but the angle of image reconstruction.

3.4 *Differentiable Simulator*
Over the past few years, there is an increasing amount of effort to build faster and more accurate turbulence simulators with the goal of maximizing the image reconstruction
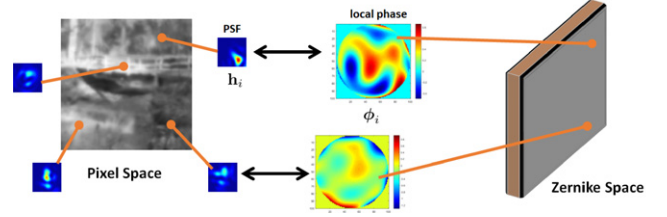
performance. One of these efforts is a series of work of random sampling in the *phase* domain. In this subsection, I briefly comment on their basic principles and highlight a few key attributes, particularly, differentiability and scalability.

Phase-over-aperture [43] (★★★★, not differentiable) One of the biggest hurdles of the spatially varying blur model $\mathcal{H}_{\boldsymbol{\theta}}^{\text{varying}}$ is that there is no simple way of constructing the spatially varying blur kernels while satisfying the turbulence physics. In addition, the pixel displacement motion field is also not a simple deformable grid. These two limitations require a new method to model the turbulence.

The idea behind the phase-over-aperture model, proposed in Ref. [43], was to recognize that the overall distortion (blur+displacement) is constructed from the Fourier magnitude square of the phase:

$$\mathbf{h}_i = |\text{Fourier}(e^{j\phi_i})|^2, \tag{24}$$

where ϕ_i is the phase function of the ith pixel. The displacement motion vector can be obtained by identifying the centroid of $\mathbf{h}_i$. Any displacement from the centroid will be marked as the pixel displacement vector. Therefore, for every pixel i, there is a one-to-one mapping from the spatial domain to the phase domain, as illustrated in Figure 10.

For any i, the phase function ϕ_i has a basis representation using the Zernike polynomials:

$$\underbrace{\phi_i}_{\text{phase function}} = \sum_{j=1}^{M} \underbrace{\alpha_{i,j}}_{\text{coefficients}} \cdot \underbrace{\mathbf{Z}_j}_{\text{Zernike polynomials}}. \tag{25}$$

Since the Zernike polynomials are known functions, the blur kernel $\mathbf{h}_i$ is constructed as soon as the coefficients $\alpha = \{\alpha_{i,j} \mid i = 1, \ldots, N, j = 1, \ldots, M\}$ become available.

The construction of the coefficient $\alpha_{i,j}$ can be done by sampling it from an enormous covariance matrix if we follow Tatarskii's assumption that the underlying random process is Gaussian [11]. In this case, by following the tedious derivations shown in Ref. [43], the coefficients can be obtained by

$$\alpha = \text{Sampling from } (\mathbb{E}[\alpha\alpha^T]), \tag{26}$$

where $\mathbb{E}[\alpha\alpha^T]$ denotes the correlation matrix. The correlation matrix can be derived from the turbulence statistics

Figure 10. Phase-over-aperture model assumes that every point spread function has a corresponding phase function. Thus, as long as we can generate the phase functions efficiently, we can construct the point spread function *without* using numerical propagation techniques.

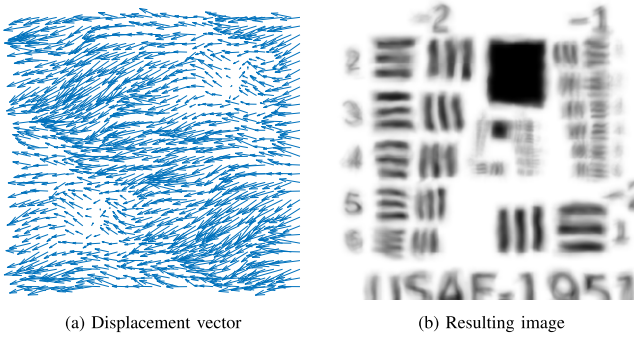(a) Displacement vector   (b) Resulting image

Figure 11. Phase-over-aperture [43]. (a) The displacement motion field, and (b) The resulting image. Note that the spatially varying blur in Figure 9 is generated using this phase-over-aperture model.

Figure 12. Phase-to-space (P2S) simulator $\mathcal{H}_{\theta}^{\text{P2S}}$ defines a mapping from the phase domain to the spatial domain using two different forms of linear representations. The mapping is implemented using a shallow neural network, and hence the model is differentiable.

where there are formulae to employ. Depending on how much turbulence statistics is utilized, α can maintain some degree of spatial correlations from one pixel in the scene to another pixel, and from one Zernike basis to another Zernike basis. If the turbulence condition is changed, then the correlation matrix will change and so the random coefficients change too.

Figure 11 shows a random motion field generated by the phase-over-aperture model, as well as the resulting image.

The bottomline of the simulator is that it turns the split-step propagation equation into a random sampling method. This gives us a model

$$\underbrace{\widehat{\mathbf{y}}_i = \mathbf{h}_i \circledast [\mathbf{t}_i \circ \mathbf{x}_i]}_{\mathcal{H}_{\theta}^{\text{phase}}(\mathbf{x}_i)}, \qquad (27)$$

where $\mathbf{t}_i$ is the pixel displacement vector that can be extracted from the first two Zernike polynomial bases. The operator $\circ$ denotes the pixel displacement operation.

CIF Trade-off for Phase-Over-Aperture Model.

- $\mathcal{E}_{\text{sim}}$: Very close to $\mathcal{H}_{\theta}^{\text{split}}$ except for extreme turbulence levels beyond the Kolmogorov regime.
- $\mathcal{E}_{\text{complexity}}$: More complex than the deformable model, but still much simpler than the split-step propagation method.
- $\mathcal{E}_{\text{recon}}$: Small. However, the usage of $\mathcal{H}_{\theta}^{\text{phase}}$ is limited to generating training datasets. No simulator in the loop because $\mathcal{H}_{\theta}^{\text{phase}}$ is not differentiable.
- $\mathcal{E}_{\text{diff}}$: Infinity. The model is not differentiable.

Phase to Space (★ ★ ★★, differentiable) [31]. The improvement brought by the phase-over-aperture model is substantial. But in order for it to become useful for image restoration, the speed needs to be further improved and the simulator needs to be differentiable.
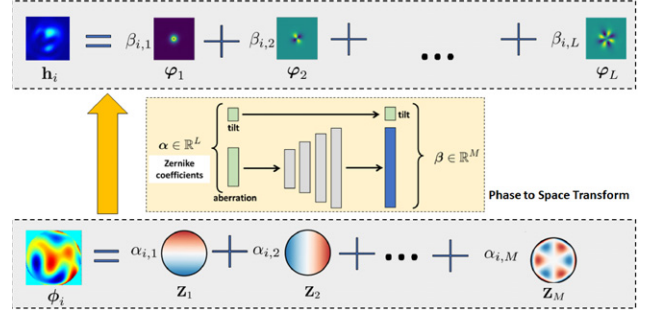
To enable a differentiable simulator while being fast, scholars recognized that the blur can be efficiently represented in a low-dimensional space using fixed and pre-defined basis functions, as follows:

$$\mathbf{h}_i = \sum_{\ell=1}^{L} \beta_{i,\ell} \varphi_{\ell}. \qquad (28)$$

Here, the functions φ_{ℓ} is the ℓth basis function of the blur, and $\beta_{i,\ell}$ is the ℓth coefficient for the ith blur kernel. A common choice of the basis functions is the Gaussian basis and its derivatives, although the authors of [31] showed that one can also perform principal component analysis (PCA) to construct the basis functions.

Why do we need to write $\mathbf{h}_i$ as a linear combination of the basis functions? The reason is that we can now represent all the $\mathbf{h}_i$'s using the coefficient vector $\beta = \{\beta_{i,\ell} \mid i = 1, \ldots, N, \ell = 1, \ldots, L\}$. Then, as long as the relationship between α and β is established, we can go back and forth between the representations.

$$\alpha \stackrel{\text{Phase to Space}}{\longrightarrow} \beta \qquad (29)$$

The forward mapping from α to β is called the phase-to-space transform (P2S) [31][2]. P2S is implemented using a very shallow neural network because the dimensionality of the input is typically $\alpha \in \mathbb{R}^{36}$ whereas that of the output is around $\beta \in \mathbb{R}^{100}$. Thus, a shallow 3-layer fully connected network is sufficient to learn the mapping. Figure 12 illustrates the conceptual diagram of P2S.

With P2S, the turbulence model is now fully differentiable. This is because the distorted pixel can be obtained through the linear equation

$$\underbrace{\widehat{\mathbf{y}}_i = \sum_{\ell=1}^{L} \beta_{i,\ell} \left[\varphi_{\ell} \circledast \mathbf{x} \right]}_{\mathcal{H}_{\theta}^{\text{P2S}}(\mathbf{x}_i)}, \qquad (30)$$

---

[2] The inverse mapping from β to α is known as the space-to-phase transform (S2P). The implementation of S2P remains an open problem, as on date.

where $\theta = \beta$. As long as β is known, the simulator only needs to perform the basis convolutions. Since $\beta \leftrightarrow \alpha$ via P2S, the simulation model is completely characterized by the Zernike coefficient α. Moreover, since the P2S model $\mathcal{H}_{\theta}^{\text{P2S}}$ is linear in β, any gradient with respect to β can be computed. Even if we need to compute the gradient with respect to α, we can do so by backpropagating the gradient through the P2S neural network (which is a three-layer fully connected network.)

CIF Trade-off for Phase-to-Space (P2S) Model.

- $\mathcal{E}_{\text{sim}}$: Same simulation error as $\mathcal{H}_{\theta}^{\text{phase}}$. P2S is just a (much) faster version.
- $\mathcal{E}_{\text{complexity}}$: Lower complexity than the phase-over-aperture model.
- $\mathcal{E}_{\text{recon}}$: Small. Since $\mathcal{H}_{\theta}^{\text{P2S}}$ is significantly faster than $\mathcal{H}_{\theta}^{\text{phase}}$, it can generate a much larger dataset. $\mathcal{H}_{\theta}^{\text{P2S}}$ can be integrated into the reconstruction model.
- $\mathcal{E}_{\text{diff}}$: Zero. The model is differentiable.

Further Improvements. There are additional points scholars proposed to improve the speed and generality of the P2S simulators. For example, in [44], it was shown that the off-diagonal blocks of the correlation matrix $\mathbb{E}[\alpha\alpha^T]$ can be truncated without hurting the performance. In [45], it was shown that with a carefully designed reformulation, the simulator can be generalized to an arbitrary turbulence profile along the path (i.e., the structure constant C_n^2 changes with respect to the distance). Both changes do not affect the differentiability of the simulator, but they make the simulator even faster and more accurate. Another modification is the recognition of the scattering/gathering forms of convolution in [46]. It was shown that for the physics to be valid, the convolution needs to be implemented in the scattering form.

3.5 *Simulator-in-Reconstruction*
An important aspect of CIF is to use simulators in reconstruction. The goal of this subsection is to explain how this can be (has been) achieved.

Simulator as Synthesizer for Training Data. The most straightforward application of the simulators is to use them to synthesize training data. Starting with a collection of clean images, the simulator generates the turbulence effect to produce a distortion-clean pair.

One often overlooked usage of a simulator is its *flexibility* in synthesizing data according to the needs of the training:

- Multiple scales: A simulator can synthesize turbulence across different resolutions. For example, for a fixed propagation distance and object size, a lower resolution image would require a smaller displacement vector and a smaller blur kernel. These images are often easier to restore. If the simulator can produce multi-resolution distortions, the overall restoration will be benefited.
- Tilt-free, blur-free, all-free: A simulator can generate tilt-only distortions, blur-only distortions, or no distor-
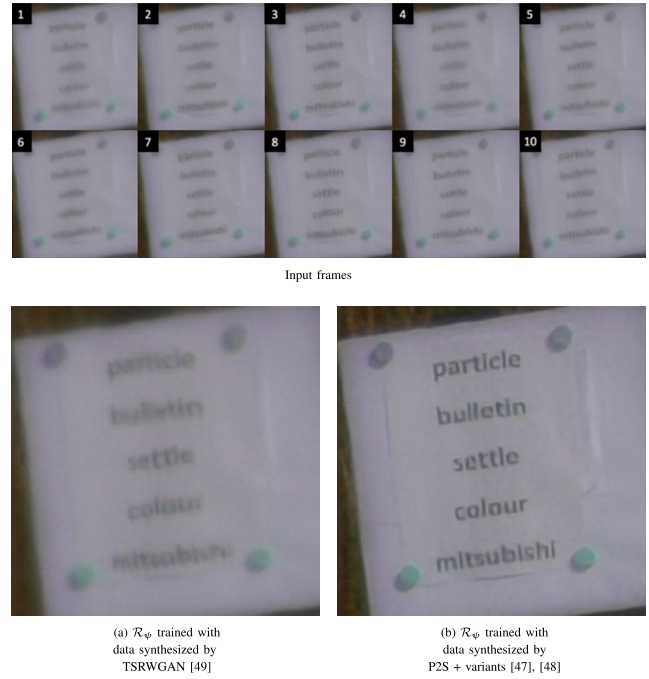

Input frames

(a) $\mathcal{R}_{\psi}$ trained with data synthesized by TSRWGAN [49]

(b) $\mathcal{R}_{\psi}$ trained with data synthesized by P2S + variants [47], [48]

Figure 13. How much difference does a simulator make? This figure shows the reconstruction result of a network trained using datasets generated by two different simulators: TSRWGAN [49] and P2S [47]. With a fixed network architecture, the effect of the simulator is evident.

tion. This provides a powerful way to *disentangle* the coupling effects of the turbulence and object movement. For approaches such as knowledge distillation or student-teacher learning, the decoupling capability offered by a simulator is the key enabler because there is no alternative ways we can train those models.

How much improvement does a good simulator offer when compared to a bad simulator? Various studies have reported a consistent observation that the difference is significant [47, 48]. Figure 13 shows a comparison between TSRWGAN [49] and a variant of the P2S [47], reported in [48]. A common neural network architecture is chosen, and it is trained using two different datasets. TSRWGAN is a more rudimentary simulator with simple deformable grid and blurs, whereas P2S is more advanced. The results shown in Fig. 13 provides a strong evidence that a better simulator indeed makes a big difference in terms of image restoration.

Simulator inside a Generative Adversarial Network. A simulator can be directly used in image restoration, for example through the generative adversarial network (GAN) [50, 51]. In the GAN setting, the simulator can be used as part of the generative branch to synthesize what a distorted image should look like. This mirrors nature where the image formation is determined by physics and the image sensor. The performance of the simulator has a direct impact on the performance of the GAN. If the simulator fails to mimic nature, then the simulated distorted image would not appear similar to the true distorted image. This adds more burden to the generator where it needs to compensate for the mismatch
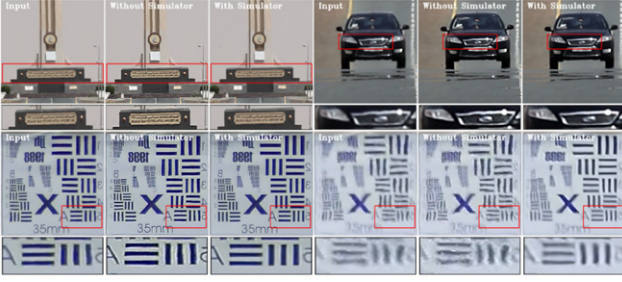
Figure 14. What happens when a simulator is not used? The image reconstruction performance is significantly worse. The figure is adopted from [52].

error caused by the simulator in addition to generating the latent unknown image. An accurate split-step propagation would not work either because it is simply too slow and it is not differentiable.

Simulator for Re-degradation Loss. Besides GAN, another approach to use simulator in the reconstruction loop is to consider the consistency loss, defined as

$$\text{Consistency}\Big(\underbrace{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{R}_{\psi}(\mathbf{y}))}_{\mathcal{H}_{\boldsymbol{\theta}}(\widehat{\mathbf{x}})},\ \mathcal{G}(\mathbf{x})\Big). \tag{31}$$

This is an additional loss put on top of the traditional reconstruction loss $\text{ReconLoss}(\widehat{\mathbf{x}}, \mathbf{x})$.

The performance of the image reconstruction depends on two factors: (i) How good is $\mathcal{H}_{\boldsymbol{\theta}}$ compared to $\mathcal{G}$? (ii) How good is $\widehat{\mathbf{x}}$ compared to the ground truth $\mathbf{x}$? Many scholars have asked why $\text{ReconLoss}(\widehat{\mathbf{x}}, \mathbf{x})$ is insufficient. The answer is that the reconstruction loss never provides any explicit knowledge about the forward model. Since $\mathcal{G}$ and $\mathcal{H}_{\boldsymbol{\theta}}$ are often ill-conditioned and so many $\mathbf{x}$ can be mapped to the same $\mathbf{y}$, the consistency loss helps the reconstruction by enforcing it not to create artifacts that cannot be explained by the forward model. The benefit of the consistency loss is supported by numerical evidence in Figure 14.

Looking Forward. The results of the latest reconstruction methods are promising. However, the full power of CIF is yet to be explored. Here are a few observations:

- **State vector θ update**. As I explained previously, the state vector update is analogous to the blur kernel estimation problem in blind deconvolution. Given a distorted image, having the ability to estimate the turbulence parameters could significantly reduce the uncertainty of the reconstruction. However, as on date, these ideas are yet to be developed.
- **State encoding**. The state vector θ is often a high dimensional vector, e.g., the collection of Zernike coefficients over the entire image. However, it is likely that θ has a low dimensional representation. Presently, there is little understanding of how these state vectors can be encoded more efficiently.
- **Bijective mapping from pixel space to embeddings**. An open problem today is the non-uniqueness of

the low-dimensional representation. Using Zernike coefficients as an example, it is possible that two sets of Zernike coefficients can give the same pixel-level image distortion. Therefore, while P2S is easy to do, the inverse mapping S2P can be significantly harder. For many problems dealing with optics through different environments, such a bijective mapping is an important technical challenge.

- **Ultra-fast simulator**. Advanced simulators such as P2S can achieve 40 frames per second for a 512-by-512 image. However, for P2S to be used as part of an iterative algorithm or used as an integral part of the reconstruction neural network, the runtime would probably need to be suppressed to a microsecond range. This is a major challenge for both hardware (GPU) and algorithm.

4. MORE EXAMPLES
In this section, I would like to elaborate more on how CIF could fit other imaging problems by discussing a few other examples in addition to the turbulence example above.

4.1 *De-raining and de-hazing*
Consider the problem of imaging through rain and haze. Like imaging through atmospheric turbulence, the exact ground truth image formation $\mathbf{y} = \mathcal{G}(\mathbf{x})$ cannot be determined exactly due to the stochastic nature of the process [53, 54].

In the literature, the degradation can be modeled in different ways [55]. For example, if the distortion is caused by **rain streak**, then

$$\mathcal{H}_{\boldsymbol{\theta}}^{\text{streak}}(\mathbf{x}) = \mathbf{x} + \mathbf{b}, \tag{32}$$

where $\mathbf{b}$ is a sparse vector representing the line streak effect of the rain. If the distortion is caused by **adherent raindrops**, [56] proposed a model

$$\mathcal{H}_{\boldsymbol{\theta}}^{\text{raindrop}}(\mathbf{x}) = (1 - \mathbf{M}) \odot \mathbf{x} + \mathbf{d}, \tag{33}$$

where $\mathbf{M}$ is a binary mask indicating whether a pixel has a raindrop, $\odot$ is the elementwise multiplication, and $\mathbf{d}$ is a sparse vector with localized scattering raindrops. If the distortion is caused by **haze and rain streak**, then the model is

$$\mathcal{H}_{\boldsymbol{\theta}}^{\text{haze}}(\mathbf{x}) = \mathbf{x} \odot \mathbf{t} + \mathbf{A}(1 - \mathbf{t}) + \mathbf{b}, \tag{34}$$

where $\mathbf{t}$ is the transmission map (see Example 2), $\mathbf{A}$ is the airlight color transformation, and $\mathbf{b}$ is a sparse vector representing the streak.

In recent years, there has been increased efforts to use neural networks to *learn* the model so that $\mathcal{H}_{\boldsymbol{\theta}}$ can be more similar to $\mathcal{G}$. For example, [57] proposed the model

$$\mathbf{y}_t = \mathcal{H}_{\boldsymbol{\theta}}^{\text{dynamic}}(\mathbf{x}_t) = \mathbf{x} + \mathbf{b}_t + \mathbf{n}_t, \tag{35}$$

where t denotes the time, and $\mathbf{n}_t \sim \text{Gaussian}(0, \sigma^2)$ is the noise vector. The rain model $\mathbf{b}_t$ is defined through some
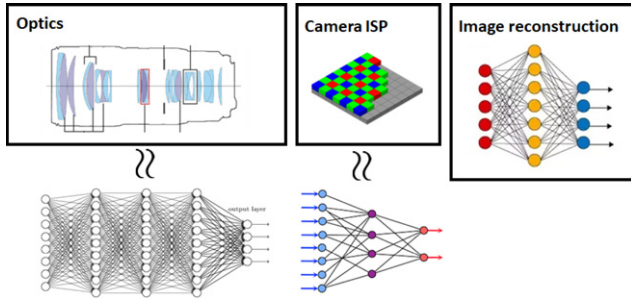
Figure 15. Differentiable optics. In a conventional camera system, there is an optical component, an image signal processing (ISP) unit which controls the exposure, color filter, etc, and an image reconstruction algorithm. Differentiable optics aims at approximating the physical optics module and the ISP with neural networks so that the entire system is differentiable. This will enable end-to-end training of the image reconstruction algorithm.

variations of the Markov process such that

$$\mathbf{b}_t = \mathcal{A}_\alpha(\mathbf{b}_{t-1}), \qquad (36)$$

where $\mathcal{A}_\alpha$ is a neural network that provides a memoryless update based on previous time stamps. Because $\mathcal{A}_\alpha$ is a neural network, it inherits several desired properties such as being differentiable. Since the simulator is co-optimized with the reconstruction algorithm, the reconstruction performance is evidently better.

Besides these examples, the general direction of imaging through adverse weather today is to inject more physics into the problem [58]. In the area of rain, snow, and fog, there is an increasing amount of high quality physics-based simulators that can simulate these optical effects [59, 60]. The usage highlights the relevance of CIF.

4.2 *Differentiable Optics*
While CIF is mostly concerned with degradation processes arising from nature, the concept can be applied to other forms of optics-algorithm co-design problems such as the one summarized in Figure 15.

(1) **Differentiable optics for lens systems** [61–64]. Traditional lens design is a standalone process where people use ray-tracing tools such as Zemax to optimize the parameters of the lenses. If one wants to design the downstream image reconstruction algorithm, these ray tracing tools, however, would be incompatible with the reconstruction. To overcome the difficulty, various methods have been proposed to approximate the true lens system $\mathcal{G}$ with neural networks $\mathcal{H}_\theta$. Since the reconstruction is usually a neural network, having a neural network $\mathcal{H}_\theta$ will give us an end-to-end differentiable camera system.

Today, $\mathcal{H}_\theta$ is mainly used to improve the image reconstruction. There is relatively little work on co-designing the lens parameters. The reason is that co-designing the lens parameters would require a method to "translate" the weights of the neural network $\mathcal{H}_\theta$ to the lens parameters. This is largely an open problem.

(2) **Metasurface design** [65, 66]. Another usage of CIF is the design of the metasurfaces. Metasurfaces are nanoscale materials where each element can be engineered to perform a specific phase operation. Compared to traditional glass-based lenses which are bulky, metasurfaces are substantially thinner while achieving a competitive optical performance. The design of metasurfaces is often performed together with the image reconstruction algorithm. This is because metasurfaces today still have many limitations in terms of chromatic aberration control and spatially varying points spread functions. Therefore, it is necessary to co-design a deconvolution algorithm.

(3) **Differentiable rendering** or computational light transport [67–69]. The problem here is more concerned with a realistic rendering of objects in computer graphics. For example, as light propagates through milk and wax, how does the image look like? Or, if the light source is located at a certain position in the room, how will the light bounce among the walls and eventually reach the camera? Because of the complexity of the actual environment and the underlying image formation process (which sometimes requires us solving partial differential equations), newer approaches attempt the problem by approximating the ground truth $\mathcal{G}$ with a neural network $\mathcal{H}_\theta$.

4.3 *Image Sensor Circuit Model*
Thus far I have been mainly talking about optics. But CIF can be extended to other components such as the circuit level modeling of image sensors from photo diodes, comparator, capacitors to the output signal.

The top row of Figure 16 shows the circuit diagram of a dynamic vision sensor (DVS) [70]. The exact signal formation process $\mathbf{y} = \mathcal{G}(\mathbf{x})$ is both stochastic and complex. However, it is possible to approximate the transient behavior using an ordinary differential equation and the probabilistic events by drawing samples from a pre-defined covariance matrix. This leads to the bottom row of Fig. 16 where there are two ordinary differential equation blocks and two autoregressive blocks. Simpler models have been proposed [71], but it was shown that the performance is not sufficient.

Besides DVS, there is also a growing interest in novel digital image sensors including photon counting devices, quanta image sensors (QIS), and single-photon avalanche diodes. Using QIS as an example, a series of 1-bit and multi-bit models have been proposed [72–74] and analyzed [75, 76], together with various image reconstruction algorithms [77–81] and applications [82, 83].

5. THOUGHTS AND DISCUSSIONS
In this section I summarize a few commonly asked questions about CIF.

Is CIF = model-based image reconstruction (MBIR)? My view is that CIF and MBIR are aiming for two different goals. In MBIR, the premise of the problem is that someone has given us a model $\widehat{\mathbf{y}} = \mathcal{H}_\theta(\mathbf{x})$. Our job is to find the best algorithm to solve for $\mathbf{x}$, by exploiting various signal priors such as sparsity or generative models. CIF, in contrast,
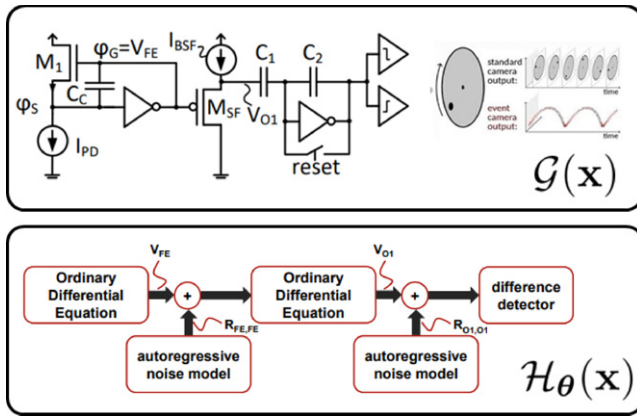
Figure 16. What happens when a simulator is not used? The image reconstruction performance will be significantly worse. The figure is adopted from [70].

primarily focuses on the design of $\mathcal{H}_\theta$. As illustrated in the previous sections, there are accurate but complex $\mathcal{H}_\theta$ and less accurate but effective $\mathcal{H}_\theta$ for the inverse solver. Constructing a meaningful $\mathcal{H}_\theta$ while maintaining the computational efficiency is what CIF is about.

Why not just use neural networks to approximate $\mathcal{G}$? With the growth of building deep neural networks to mimic the optics, we might be tempted to think that a good CIF simulator must be a neural network (so that everything is differentiable). I personally think that this is not the best (nor the only) direction. While I completely acknowledge the power of a neural network, I do not think today's neural networks have advanced to the stage that it can perform every task without using a much larger model. For some specialized tasks such as solving an ordinary differential equation, a neural network could offer a powerful approximation. But for equations such as Fresnel propagation, Fourier transforms are much more efficient.

I envision that future CIF simulators will most likely be a hybrid model where neural networks are used as one of the building blocks to complete some specific sub-tasks. Differentiability can be ensured *without* a neural network. For example, in the turbulence case study described above, the differentiability is enabled by a different representation of the phase function. Even for tasks such as ray tracing, it will be far more interesting (and impactful) to derive new equations that preserves differentiability without using automatic differentiation of computational graph of any sort.

Is sensor-algorithm co-optimization always needed? As I follow past few year's of publications in computational imaging, I observe a trend that whenever we see a sensor and an algorithm, it will be sensor-algorithm co-optimization. I can see the necessity of co-optimization if the goal is to maximize the system's performance unconditionally. However, from a practical point of view, we should not forget about the feasibility and physical constraints. In a recent paper [84], it was shown that co-optimization brings negligible benefits to the actual performance in some specific

problems. This counter example is perhaps a good reminder to us about the reality.

6. CONCLUSION

Computational imaging is the intersection of image acquisition, signal prior, and numerical algorithm. Forty years ago when we were still in the beginning of solving inverse imaging problems, our attention was mostly spent on developing better and more powerful priors (i.e., regularization functions such as ℓ_1 norm, total variation, Markov random field, etc) together with faster numerical algorithms (e.g., gradient descent, alternating minimization, operator splitting, etc). As we continue to advance computational imaging in 2023, it is perhaps time to rethink about the role of the *forward model* that describes the image acquisition process.

This article describes the concept of computational image formation (CIF). CIF highlights the choice of a simulator $\mathcal{H}_\theta$ that approximates the true image formation process $\mathcal{G}$. Unlike a physics simulator whose goal is to match $\mathcal{G}$ unconditionally, the simulator $\mathcal{H}_\theta$ in CIF needs to maximize the image reconstruction performance while matching $\mathcal{G}$ up to some level. Moreover, the simulator needs to be very fast so that it can be used to generate data, and it needs to be differentiable so that it can be used in the reconstruction loop. Several examples are elaborated to explain CIF.

We all stand on the shoulder of giants. CIF is no exception. It is a concept summarizing decades of collective efforts of the computational imaging community. As we look forward to the future of imaging, it is I envision that simulators will play an unprecedented role in the deep learning era.

REFERENCES

[1] R. G. Frisius, *De Radio Astronomico et Geometrico Liber* (ap. Gul Cavellat, 1558), ISBN-10: 1166638855.

[2] C. Bouman, *Foundations of Computational Imaging: A Model-Based Approach* (SIAM, Philadelphia, PA, 2022), ISBN: 978-1-61197-712-7.

[3] A. Bhandari, A. Kadambi, and R. Raskar, *Computational Imaging* (MIT Press, Cambridge, MA, 2022).

[4] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2009), pp. 1956–1963.

5 R. Fattal, "Single image dehazing," ACM Trans. Graph. (TOG) **27**, 1–9 (2008).

6 M. C. W. van Rossum and T. M. Nieuwenhuizen, "Multiple scattering of classical waves: Microscopy, mesoscopy, and diffusion," Rev. Mod. Phys. **71**, 313–371 (1999).

7 H. Koschmieder, "Theorie der horizontalen sichtweite," Beitrage zur Physik der freien Atmosphare **12**, 33–53 (1924).

8 A. Gnanasambandam, A. M. Sherman, and S. H. Chan, "Optical adversarial attack," *IEEE/CVF Int'l. Conf. on Computer Vision (ICCV) Workshops* (IEEE, Piscataway, NJ, 2021), pp. 92–101.

9 Y. Sanghvi, Z. Mao, and S. H. Chan, "Structured kernel estimation for photon-limited deconvolution," *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2023), pp. 9863–9872.

10 J. Ding, V. Tarokh, and Y. Yang, "Model selection techniques: An overview," IEEE Signal Process. Mag. **35**, 16–34 (2018).

11 V. I. Tatarskii, *Wave Propagation in a Turbulent Medium* (Dover Publications, New York, NY, 1961).

12 M. C. Roggemann and B. M. Welsh, *Imaging through Atmospheric Turbulence* (Taylor & Francis, Oxfordshire, 1996).

13 J. D. Schmidt, *Numerical Simulation of Optical Wave Propagation: With Examples in MATLAB* (SPIE Press, Bellingham, WA, 2010).

14 J. W. Goodman, *Introduction to Fourier Optics*, 3rd ed. (Roberts and Company, Greenwood Village, CO, 2005).

15 J. W. Goodman, *Statistical Optics*, 2nd ed. (John Wiley and Sons Inc., Hoboken, NJ, 2015).

16 A. N. Kolmogorov, "The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers," Dokl. Akad. Nauk SSSR **30**, 301–305 (1941).

17 U. Frisch, *Turbulence: The Legacy of A. N. Kolmogorov* (Cambridge University Press, Cambridge, 1995).

18 R. G. Lane, A. Glindemann, and J. C. Dainty, "Simulation of a Kolmogorov phase screen," Waves Random Media **2**, 209–224 (1992).

19 J. P. Bos, M. C. Roggemann, and V. S. R. Gudimetla, "Anisotropic non-kolmogorov turbulence phase screens with variable orientation," Appl. Opt. **54**, 2039–2045 (2015).

20 K. R. Leonard, J. Howe, and D. E. Oxford, "Simulation of atmospheric turbulence effects and mitigation algorithms on stand-off automatic facial recognition," Proc. SPIE **8546**, 1–18 (2012).

21 G. Potvin, J. Luc Forand, and D. Dion, "A simple physical model for simulating turbulent imaging," Proc. SPIE **8014**, 80140Y (2011).

22 R. C. Hardie, J. D. Power, D. A. LeMaster, D. R. Droege, S. Gladysz, and S. Bose-Pillai, "Simulation of anisoplanatic imaging through optical turbulence using numerical wave propagation with new validation analysis," Opt. Eng. **56**, 071502 (2017).

23 J. P. Bos and M. C. Roggemann, "Technique for simulating anisoplanatic image formation over long horizontal paths," Opt. Eng. **51**, 101704 (2012).

24 A. Schwartzman, M. Alterman, R. Zamir, and Y. Y. Schechner, "Turbulence-indueced 2D correlated image distortion," *Proc. Int'l. Conf. on Computational Photography* (IEEE, Piscataway, NJ, 2017), pp. 1–12.

25 M. A. Vorontsov and V. Kolosov, "Target-in-the-loop beam control: Basic considerations for analysis and wave-front sensing," J. Opt. Soc. Am. A **22**, 126–141 (2005).

26 S. L. Lachinova, M. A. Vorontsov, V. V. Dudorov, V. V. Kolosov, and M. T. Valley, "Anisoplanatic imaging through atmospheric turbulence: Brightness function approach," Proc. SPIE **6708** (2007).

27 M. C. Roggemann, B. M. Welsh, D. Montera, and T. A. Rhoadarmer, "Method for simulating atmospheric turbulence phase effects for multiple time slices and anisoplanatic conditions," Appl. Opt. **34**, 4037–4051 (1995).

28 E. Repasi and R. Weiss, "Analysis of image distortions by atmospheric turbulence and computer simulation of turbulence effects," Proc. SPIE **6941**, 69410S (2008).

29 G. Potvin, "Linear perturbation model for simulating imaging through weak turbulence," Opt. Eng. **59**, 034105 (2020).

30 R. C. Hardie, M. A. Rucci, A. J. Dapore, and B. K. Karch, "Block matching and wiener filtering approach to optical turbulence mitigation and its application to simulated and real imagery with quantitative error analysis," Opt. Eng. **56**, 071503 (2017).

31 Z. Mao, N. Chimitt, and S. H. Chan, "Accelerating atmospheric turbulence simulation via learned phase-to-space transform," *IEEE/CVF*

Int'l. Conf. on Computer Vision (IEEE, Piscataway, NJ, 2021), pp. 14759–14768.

32 M. Shimizu, S. Yoshimura, M. Tanaka, and M. Okutomi, "Super-resolution from image sequence under influence of hot-air optical turbulence," *IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2008), pp. 1–8.

33 Y. Mao and J. Gilles, "Non rigid geometric distortions correction - application to atmospheric turbulence stabilization," Inverse Probl. Imaging **3**, 531–546 (2012).

34 G. Gilles and S. Osher, "Wavelet burst accumulation for turbulence mitigation," J. Electron. Imaging **25**, 033003 (2016).

35 Y. Lou, S. H. Kang, S. Soatto, and A. Bertozzi, "Video stabilization of atmospheric turbulence distortion," Inverse Prob. Imaging **7**, 839–861 (2013).

36 X. Zhu and P. Milanfar, "Removing atmospheric turbulence via space-invariant deconvolution," IEEE Trans. Pattern Anal. Mach. Intell. **35**, 157–170 (2013).

37 C. P. Lau, Y. H. Lai, and L. M. Lui, "Restoration of atmospheric turbulence-distorted images via RPCA and quasiconformal maps," J. Inverse Probl. **35**, 1–33 (2019).

38 C. P. Lau and L. M. Lui, "Subsampled turbulence removal network," Math. Comput. Geom. Data **1**, 1–33 (2021).

39 Z. Mao, N. Chimitt, and S. H. Chan, "Image reconstruction of static and dynamic scenes through anisoplanatic turbulence," IEEE Trans. Comput. Imaging **6**, 1415–1428 (2020).

40 N. Anantrasirichai, A. Achim, N. G. Kingsbury, and D. R. Bull, "Atmospheric turbulence mitigation using complex wavelet-based fusion," IEEE Trans. Image Process. **22**, 2398–2408 (2013).

41 N. Anantrasirichai, A. Achim, and D. Bull, "Atmospheric turbulence mitigation for sequences with moving objects using recursive image fusion," *IEEE Int'l. Conf. on Image Processing* (IEEE, Piscataway, NJ, 2018), pp. 2895–2899.

42 S. H. Chan, "Tilt-then-blur or blur-then-tilt? clarifying the atmospheric turbulence model," IEEE Signal Process. Lett. **29**, 1833–1837 (2022).

43 N. Chimitt and S. H. Chan, "Simulating anisoplanatic turbulence by sampling intermodal and spatially correlated Zernike coefficients," Opt. Eng. **59**, 083101 (2020).

44 N. Chimitt, X. Zhang, Z. Mao, and S. H. Chan, "Real-time dense field phase-to-space simulation of imaging through atmospheric turbulence," IEEE Trans. Comput. Imaging **8**, 1159–1169 (2022).

45 N. Chimitt and S. H. Chan, "Anisoplanatic optical turbulence simulation for near-continuous c_n^2 profiles without wave propagation," SPIE Opt. Eng. **62**, 078103 (2023).

46 N. Chimitt, X. Zhang, Y. Chi, and S. H. Chan, "Scattering and gathering for spatially varying blurs," Preprint, arXiv:2303.05687 (2023).

47 Z. Mao, A. Jaiswal, Z. Wang, and S. H. Chan, "Single frame atmospheric turbulence mitigation: A benchmark study and a new physics-inspired transformer model," *European Conf. on Computer Vision* (Springer, Cham, 2022), pp. 430–446.

48 X. Zhang, Z. Mao, N. Chimitt, and S. H. Chan, "Imaging through the atmosphere using turbulence mitigation transformer," Available online: https://arxiv.org/abs/2207.06465. (2022).

49 D. Jin, Y. Chen, Y. Lu, J. Chen, P. Wang, Z. Liu, S. Guo, and X. Bai, "Neutralizing the impact of atmospheric turbulence on complex scene imaging via deep learning," Nature Mach. Intell. **3**, 876–884 (2021).

50 C. P. Lau, H. Souri, and R. Chellappa, "Atfacegan: Single face semantic aware image restoration and recognition from atmospheric turbulence," IEEE Trans. Biometrics Behav. Identity Sci. **3**, 240–251 (2021).

51 B. Y. Feng, M. Xie, and C. A. Metzler, "Turbugan: An adversarial learning approach to spatially-varying multiframe blind deconvolution with applications to imaging through turbulence," IEEE J. Sel. Areas Inf. Theory **3**, 543–556 (2022).

52 A. Jaiswal, X. Zhang, S. H. Chan, and Z. Wang, "Physics-driven turbulence image restoration with stochastic refinement," *IEEE Int'l. Conf. Computer Vision (ICCV)* (IEEE, Piscataway, NJ, 2023), pp. 12170–12181.

53 A. Cantor, "Optics of the atmosphere–scattering by molecules and particles," IEEE J. Quantum Electron. **14**, 698–699 (1978).

54 K. Garg and S. K. Nayar, "When does a camera see rain?," *IEEE Int'l. Conf. on Computer Vision (ICCV)* (IEEE, Piscataway, NJ, 2005), Vol. 2, pp. 1067–1074.

55 S. Li, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda, R. H. Junior, R. Cesar-Junior, J. Zhang, X. Guo, and X. Cao, "Single image deraining: A comprehensive benchmark analysis," *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2019), pp. 3833–3842.

56 S. You, R. T. Tan, R. Kawakami, Y. Mukaigawa, and K. Ikeuchi, "Adherent raindrop modeling, detection and removal in video," IEEE Trans. Pattern Anal. Mach. Intell. **38**, 1721–1733 (2016).

57 Z. Yue, J. Xie, Q. Zhao, and D. Meng, "Semi-supervised video deraining with dynamical rain generator," *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2021), pp. 642–652.

58 A. Kadambi, C. de Melo, C.-J. Hsieh, M. Srivastava, and S. Soatto, "Incorporating physics into data-driven computer vision," Nature Mach. Intell. **5**, 572–580 (2023).

59 Y. Ba, H. Zhang, E. Yang, A. Suzuki, A. Pfahnl, C. C. Chandrappa, C. M. de Melo, S. You, S. Soatto, A. Wong, and A. Kadambi, "Not just streaks: Towards ground truth for single image deraining," in *Computer Vision – ECCV 2022. ECCV 2022*, edited by S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Lecture Notes in Computer Science (Springer, Cham, 2022), Vol. 13667, pp. 723–740.

60 H. Zhang, Y. Ba, E. Yang, V. Mehra, B. Gella, A. Suzuki, A. Pfahnl, C. C. Chandrappa, A. Wong, and A. Kadambi, "Weatherstream: Light transport automation of single image deweathering," *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2023), pp. 13499–13509.

61 E. Tseng, A. Mosleh, F. Mannan, K. St-Arnaud, A. Sharma, Y. Peng, A. Braun, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide, "Differentiable compound optics and processing pipeline optimization for end-to-end camera design," ACM Trans. Graph. (TOG) **40**, 18.1–18.19 (2021).

62 C. Wang, N. Chen, and W. Heidrich, "dO: A differentiable engine for deep lens design of computational imaging systems," IEEE Trans. Comput. Imaging **8**, 905–916 (2022).

63 N. Chen, L. Cao, T.-C. Poon, B. Lee, and E. Y. Lam, "Differentiable imaging: A new tool for computational optical imaging," Adv. Phys. Res. **2**, 2200118 (2023).

64 V. Sitzmann, S. Diamond, Y. Peng, X. Dun, S. Boyd, W. Heidrich, F. Heide, and G. Wetzstein, "End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging," ACM Trans. Graphics (TOG) **37**, 114.1–114.13 (2018).

65 D. S. Hazineh, S. W. D. Lim, Z. Shi, F. Capasso, T. Zickler, and Q. Guo, "D-flat: A differentiable flat-optics framework for end-to-end metasurface visual sensor design," Preprint, arXiv:2207.14780 (2022).

66 W. T. Chen, A. Y. Zhu, V. Sanjeev, M. Khorasaninejad, Z. Shi, E. Lee, and F. Capasso, "A broadband achromatic metalens for focusing and imaging in the visible," Nature Nanotechnol. **13**, 220–226 (2018).

67 A. Velten, D. Wu, B. Masia, A. Jarabo, C. Barsi, C. Joshi, E. Lawson, M. Bawendi, D. Gutierrez, and R. Raskar, "Imaging the propagation of light through scenes at picosecond resolution," Commun. ACM **59**, 79–86 (2016).

68 M. O'Toole and K. N. Kutulakos, "Optical computing for fast light transport analysis," ACM Trans. Graph. (TOG) **29**, 164.1–164.12 (2010).

69 I. Gkioulekas, A. Levin, F. Durand, and T. Zickler, "Micron-scale light transport decomposition using interferometry," ACM Trans. Graph. (TOG) **34**, 37.1–37.14 (2015).

70 A. Suess, M. Guo, R. Jiang, X. Mou, Q. Huang, W. Yang, and S. Chen, "Physical modeling and parameter extraction for event-based vision sensors," *Int'l. Image Sensor Workshop (IISW)* (IISW, Edinburgh, 2023), pp. R5.17–R5.24.

71 X. Mou, K. Feng, A. Yi, S. Wang, H. Chen, X. Hu, M. Guo, S. Chen, and A. Suess, "Accurate event simulation using high-speed video," Electron. Imaging **34**, 242-1–242-1 (2022).

72 S. H. Chan, "What does a one-bit quanta image sensor offer?," IEEE Trans. Comput. Imaging **8**, 770–783 (2022).

73 O. A. Elgendy and S. H. Chan, "Optimal threshold design for Quanta Image Sensor," IEEE Trans. Comput. Imaging **4**, 99–111 (2018).

74 A. Gnanasambandam and S. H. Chan, "HDR imaging with Quanta Image Sensors: Theoretical limits and optimal reconstruction," IEEE Trans. Comput. Imaging **6**, 1571–1585 (2020).

75 A. Gnanasambandam and S. H. Chan, "Exposure-referred signal-to-noise ratio," IEEE Trans. Comput. Imaging **8**, 561–575 (2022).

76 S. H. Chan, "On the insensitivity of bit density to read noise in one-bit quanta image sensors," IEEE Sensors J. **23**, 3666–3674 (2023).

77 S. H. Chan, O. A. Elgendy, and X. Wang, "Images from bits: Non-iterative image reconstruction for Quanta Image Sensors," MDPI Sensors **16**, 1961 (2016).

78 Y. Chi, A. Gnanasambandam, V. Koltun, and S. H. Chan, "Dynamic low-light imaging with quanta image sensors," *European Conf. on Computer Vision* (Springer, Cham, 2020), pp. 122–138.

79 A. Gnanasambandam, O. Elgendy, J. Ma, and S. H. Chan, "Megapixel photon-counting color imaging using Quanta Image Sensor," OSA Optics Express **27**, 17298–17310 (2019).

80 J. H. Choi, O. A. Elgendy, and S. H. Chan, "Image reconstruction for Quanta Image Sensors using deep neural networks," *IEEE Int'l. Conf. Acoustics, Speech, and Signal Processing* (IEEE, Piscataway, NJ, 2018), pp. 6543–6547.

81 O. A. Elgendy, A. Gnanasambandam, S. H. Chan, and J. Ma, "Low-light demosaicking and denoising for small pixels using learned frequency selection," IEEE Trans. Comput. Imaging **7**, 137–150 (2021).

82 A. Gnanasambandam and S. H. Chan, "Image classification in the dark using Quanta Image Sensors," *European Conf. on Computer Vision* (Springer, Cham, 2020), pp. 484–501.

83 C. Li, X. Qu, A. Gnanasambandam, O. A. Elgendy, J. Ma, and S. H. Chan, "Photon-limited object detection using non-local feature matching and knowledge distillation," *IEEE/CVF Int'l. Conf. on Computer Vision Workshops (ICCVW)* (IEEE, Piscataway, NJ, 2021), pp. 3959–3970.

84 X. Qu, Y. Chi, and S. H. Chan, "Spatially varying exposure with 2-by-2 multiplexing: Optimality and universality," Preprint, arXiv:2306.17367 (2023).