Characterizing Atmospheric Turbulence and Removing Distortion in Long-range Imaging

by

Cameron Whyte

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Arts

Approved April 2021 by the
Graduate Supervisory Committee:

Malena Espanol, Co-Chair
Suren Jayasuriya, Co-Chair
Gil Speyer

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Atmospheric turbulence distorts the path of light passing through the air. When capturing images at long range, the effects of this turbulence can cause substantial geometric distortion and blur in images and videos, degrading image quality. These become more pronounced with greater turbulence, scaling with the refractive index structure constant, C_n^2. Removing effects of atmospheric turbulence in images has a range of applications from astronomical imaging to surveillance. Thus, there is great utility in transforming a turbulent image into a "clean image" undegraded by turbulence. However, as the turbulence is space- and time-variant and statistically random, no closed-form solution exists for a function that performs this transformation. Prior attempts to approximate the solution include spatio-temporal models and lucky frames models, which require many images to provide a good approximation, and supervised neural networks, which rely on large amounts of simulated or difficult-to-acquire real training data and can struggle to generalize.

The first contribution in this thesis is an unsupervised neural-network-based model to perform image restoration for atmospheric turbulence with state-of-the-art performance. The model consists of a grid deformer, which produces an estimated distortion field, and an image generator, which estimates the distortion-free image. This model is transferable across different datasets; its efficacy is demonstrated across multiple datasets and on both air and water turbulence.

The second contribution is a supervised neural network to predict C_n^2 directly from the warp field. This network was trained on a wide range of C_n^2 values and estimates C_n^2 with relatively good accuracy. When used on the warp field produced by the unsupervised model, this allows for a C_n^2 estimate requiring only a few images without any prior knowledge of ground truth or information about the turbulence.

DEDICATION

To my parents and to my fiancée.

ACKNOWLEDGEMENTS

I would first like to thank Dr. Suren Jayasuriya for his continued investment as my mentor and co-chair. He has readily shared with me his knowledge and experience in computer vision, machine learning, and research more broadly. His mentorship and guidance have been invaluable to me, and I owe a great debt of gratitude to his consistent and patient commitment to academic development.

I would also like to thank my co-chair Dr. Malena Espanol. I greatly appreciate her willingness to co-chair my thesis, as well as the valuable mathematical background she has offered to the project. She has offered kind and constructive feedback to ensure the thesis meets an acceptable standard of both rigor and clarity. Additionally, I wish to thank Dr. Gil Speyer for his consistent guidance and support throughout my research career, and for once again joining my thesis committee.

The third chapter of this thesis is a slightly-revised version of a manuscript for which I am third author, and it relies on the significant contributions of researchers at Louisiana State University: Dr. Nianyi Li and Simron Thapa were responsible for developing the model, running the majority of the experiments, and writing the bulk of the manuscript presented in the chapter, while Dr. Jinwei Ye provided direction for the project and also contributed to the manuscript. Albert Reed also offered wisdom and insight to the project, and specifically gave me helpful guidance on matters of deep learning and differentiable programming. Ripon Kumar Saha provided additional data for the turbulence model, as well as useful feedback to me. In the third chapter, I contributed the turbulent long-range and hot-air video datasets that were captured with the Nikon Coolpix P1000 camera. I also performed the ablation studies involving varying number of input images and bandwidth-related scale factor. Finally, I offered edits and revisions to the manuscript, which we have submitted to ICCV 2021 for peer review and publication.

The Flinn Foundation has generously funded me through the course of both of my degrees at Arizona State University, and I am immensely grateful for the opportunities and flexibility they have afforded me.

Finally, I would like to thank my parents, whose encouragement and care has enabled me to become the student and researcher I am today, and my fiancée, Nicole, whose moral support has helped me get through the various challenges I have faced over the course of this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Turbulence is pervasive in Earth's atmosphere and bodies of water. When a ray of light passes through a turbulent medium, its path is distorted. When attempting to image through turbulence, this phenomenon must be considered because has an effect on the final image. Light enters the camera aperture at a perturbed angle as a result of this distortion. This angle changes the relative position of a given point in the resulting image captured by the camera, resulting in a geometrically distorted image. Blur also results from the dispersal of light waves emitted from a given source as they pass through the turbulent medium. This combination of geometric distortion and blur can make objects indistinct and unrecognizable, can be confused for motion, and ultimately results in substantially degraded images. The effects are exacerbated with increased turbulence. Although air and water turbulence effects share certain visual similarities, they are described by different models.

Being able to remove the effects of turbulence - to restore an image free of turbulence-induced distortion and blur - has great utility. Turbulence effects are most pronounced at long distances, so astronomical imaging, often requiring spatial precision and detail, would benefit greatly from atmospheric turbulence removal, as would aerial reconnaissance and surveillance. However, removal of atmospheric turbulence is not a simple task. In fact, physicist Richard Feynman is quoted as saying "Turbulence is the most important unsolved problem of classical physics". Suppose some function F exists that transforms a turbulence-free image into the turbulent image that is actually received by the camera. One could simply invert the function F and run find that the turbulence-free image is F' (turbulent image). However, no

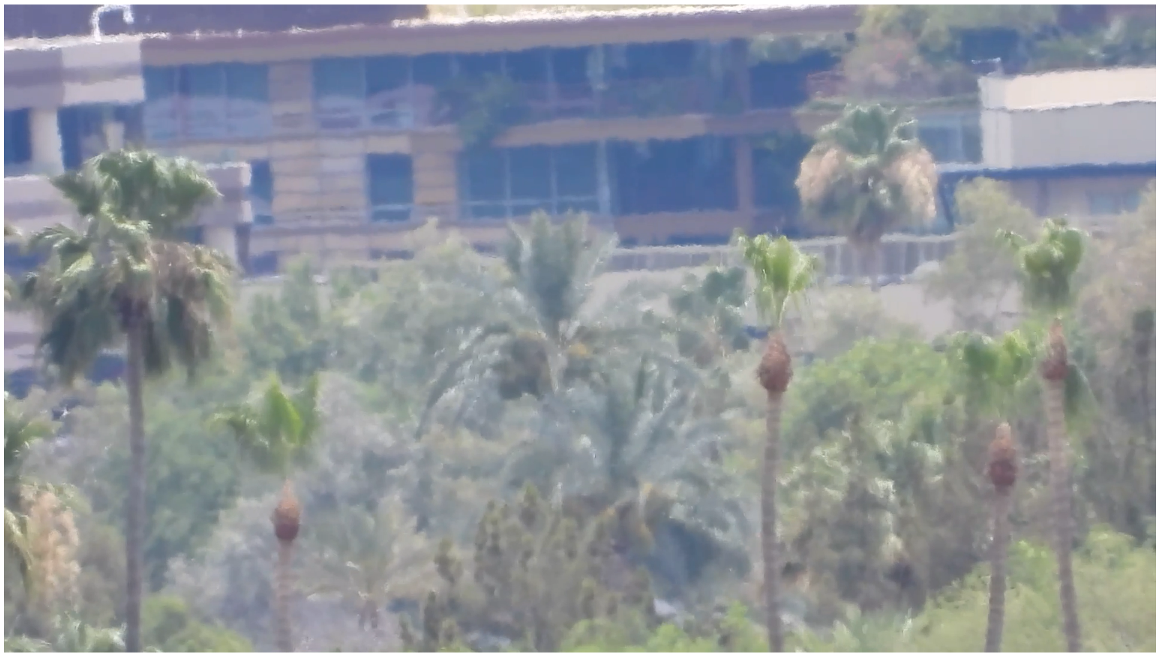


Figure 1.1: A real example of a turbulent image.

known solution exists for F or F'. Turbulence is spatio-temporally variant - in effect, the distortion and blur at a certain point is different than another point in the given image, and different at the same point in space in a different image in time - and statistically random. Thus, all methods aiming to estimate the turbulence-free image are still imperfect estimators.

A variety of models exist to accomplish this task; some are physics-based approximators of F, others apply algorithms that splice together optimal regions of multiple images, and more recently, some apply machine learning methods to approximate a solution. In this thesis, we will review some existing methods, as well as provide a method to address the unique challenges of turbulence and limitations in existing models.

Also important is the estimation of the strength of turbulence, particularly air turbulence. For instance, knowing the strength of turbulence can allow anticipation of performance degradation in electromagnetic systems. Methods exist to empirically measure the strength of turbulence, but they most frequently involve measuring instruments that can be prohibitively expensive. Methods to estimate strength of turbulence without prior sensor information about turbulent conditions are limited. Being able to estimate turbulence strength solely from images would circumvent the necessity of turbulence condition measurements, and allow determination of turbulent conditions in both the present and past. We aim to also address this problem in this thesis.

1.1 Our Contribution

In this thesis, we propose an unsupervised model that leverages convolutional neural networks to produce a distortion-free image. Our model consists of a grid deformer, which estimates the distortion field for a turbulent image, and an image generator, which directly estimates the distortion-free image; these components learn in tandem to ultimately produce a more optimal distortion-free image. We demonstrate the model achieves state-of-the-art efficacy at removing both air and water turbulence from images, while requiring relatively few input images.

In addition, we provide a method that effectively estimates the refractive index structure constant, C_n^2, which describes the strength of turbulence. We used a turbulence simulator developed by Schwartzman *et al.* to produce a dataset of 5,000 warp fields with varying C_n^2 values. We then trained a supervised convolutional neural network on this dataset to predict C_n^2 directly from the warp field. The network can be used on the warp field produced by our model, allowing for a fairly accurate C_n^2 estimate without any underlying knowledge of either the ground truth image or the turbulent conditions. The network can also be employed to estimate C_n^2 in any circumstance where a warp field is available. To our knowledge, this is the first model that estimates C_n^2 directly from warp fields.

Chapter 2

BACKGROUND

Here we provide a brief overview of turbulence and the considerations when imaging through turbulence. We then discuss supervised and unsupervised machine learning and artificial neural networks. We review prior work aiming to remove the effects of air and water turbulence, and lastly, we present our contribution.

2.1 Imaging through Turbulence

Fluid flow can be either laminar or turbulent. A fluid exhibiting laminar flow moves in layers that may flow at different velocities, but generally slide past other layers smoothly without currents perpendicular to the primary direction of flow. In contrast, a fluid exhibiting turbulent flow moves more chaotically. Rather than layers moving smoothly in a primary direction, eddies and recirculation are pervasive, and the fluid's movements appear to be random. The Navier-Stokes equations can be used to describe describe the motion of fluids. These equations and their relationship with turbulence are explored extensively in [15].

Turbulence causes the refractive index $n(\mathbf{r}, t)$ to fluctuate randomly in each location $\mathbf{R}$ and at each time t [69]. A random field in space-time is formed by $n(\mathbf{r}, t)$, which can be characterized for two distinct spatial locations $\mathbf{r_1}$ and $\mathbf{r_2}$ by the structure function shown below [69]:

$$D(\mathbf{r_1}, \mathbf{r_2}, t) = \langle |n(\mathbf{r_1}, t) - n(\mathbf{r_2}, t)|^2 \rangle$$

For isotropic turbulence (i.e. the structure function is constant over t, $D(\mathbf{r_1}, \mathbf{r_2}) = D(\mathbf{r})$, and structure function is spherically symmetric such that $D(\mathbf{r}) = D(r)$), the

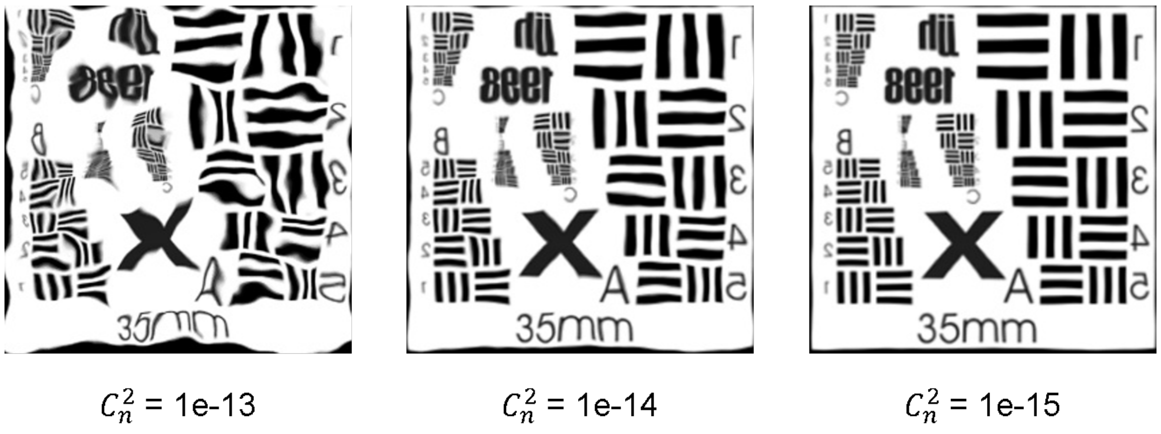


Figure 2.1: Images with simulated turbulence at different C_n^2 values.

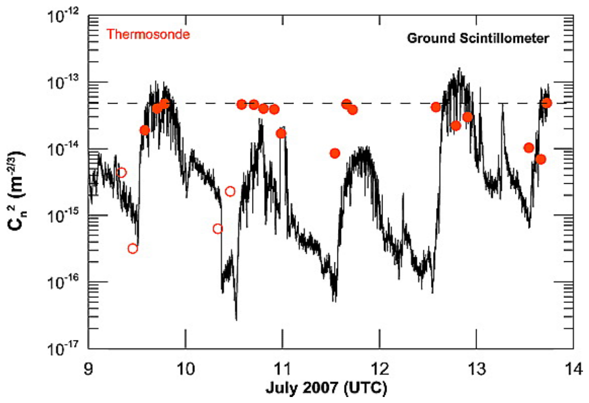


Figure 2.2: C_n^2 surface measurements at Fort Polk, July 2007, as found in [56]. Each tick on the horizontal axis marks a day.

structure function is defined by the following [69]:

$$D(r) = C_n^2 r^{2/3}$$

where C_n^2 is a constant representing the strength of turbulence (see the following section). When turbulence is non-homogeneous, C_n^2 is a function of absolute location.

The refractive index structure constant, C_n^2, is commonly used to provide a measure of the strength of atmospheric turbulence [53]. Values for the constant typically range from 10^{-16} to 10^{-12} m$^{-2/3}$ in the atmospheric surface layer [71]. Values of greater than 10^{-13} m$^{-2/3}$ indicating higher turbulence with marked affects such as waviness and blur; values below 10^{-15} m$^{-2/3}$ indicate lower turbulence. Differences varies locally and is dependent on the temperature, humidity, and wind velocity at any given point in space. Various devices exist to estimate C_n^2, including scintillometers and sonic anemometers; these devices are often prohibitively expensive, so not much C_n^2 data is widely available [77]. An example of C_n^2 measures over time at a given location is given in Fig. 2.2, and qualitative examples of different C_n^2 values are given in Fig. 2.1.

As an electromagnetic wave, light undergoes random amplitude and phase fluctuations when passing through a turbulent medium, such as air or water [69]. The angle of arrival (abbreviated AoA or AA) of the light incident at a camera is determined by the perturbed phase. The projected location of a given point in a scene is dependent on this angle of arrival. When considering the propagation of spherical waves, we can derive the following equation for the variance of the angles of arrival:

$$\langle \alpha^2 \rangle = 2.914 D^{-1/3} \int_0^L C_n^2(z) (\frac{z}{L})^{5/3} dz$$

where L is the distance between the scene and the camera, and D is the camera aperture diameter.

Air turbulence distortion is caused by the constantly changing refractive index field of the air flow. It typically occurs when imaging through long-range atmospheric turbulence or short-range hot air turbulence (e.g., fire flames, vapor streams). The motion of such turbulence is often swift. The turbulent images thus exhibit high frequency distortions. Water turbulence distortion, in contrast, is induced by the refraction of light at the water-air interface. The distortion is therefore highly relevant to the water surface geometry (e.g., the depth and normal field). Although water turbulence might not be as fast evolving as air turbulence, it causes more drastic geometric distortions in the turbulent images. As with the geometric distortion, blur induced by turbulence is also randomly spatially- and time-varying [86].

2.2 Machine Learning

Machine learning (ML) most generally describes a computer algorithm that improves itself by learning from data. ML is a foundational component of artificial intelligence, and underpins most AI algorithms by enabling them to effectively perform a task. The applications of ML vary vastly, and ML is becoming increasingly widespread and increasingly important [32]. Here we discuss the differences between supervised and unsupervised learning and provide a particular emphasis on the class of machine learning models called artificial neural networks.

Machine learning algorithms can be divided into several key categories based on the type of learning; among these are supervised learning and unsupervised learning [18]. Consider a machine learning algorithm that receives a sequence of input data $x_1, ..., x_n$ from which it will learn. These inputs may be images, spectral data, words, or other types of input. The desired output from the algorithm is some sequence $y_1, ..., y_n$ where x_i corresponds to each y_i for $i \in \mathbb{N}$. Each y_i may be a class label, a score, or even data structured in a similar format as x_i. If the algorithm is a supervised machine learning algorithm, $y_1, ..., y_n$ must be known and given to the algorithm. The algorithm then adjusts its parameters so as to optimally replicate the correct y for any given x. Algorithms such as linear regression, logistic regression, support vector machines, and K-nearest neighbors are supervised algorithms. On the other hand, a supervised algorithm relies on input data and does not require labels. Principal component analysis and cluster analysis are key unsupervised algorithms.

Artificial neural networks (also referred to as neural networks, neural nets, ANNs, or NNs) are an important class of machine learning models capable of performing relatively complex tasks. The advent of the backpropagation algorithm allowed for the effective training of multilayer neural networks [79]. ANNs have seen more widespread use in recent years, coinciding with increasing computational bandwidth, optimization of graphics processing units for ML use, and greater prevalence of distributed computing. Various types of ANNs have made particularly notable impacts on difficult challenges, including image classification, speech recognition and synthesis, and medical diagnosis and prognosis.

As their name suggests, artificial neural networks are inspired by the central nervous system's network of neurons, which passes information through electrical signals and is capable of learning highly complex information [23]. Commonly, ANNs are arranged in layers, each with a number of artificial neurons, also known as nodes. In a multilayer perceptron (MLP), each node has a weighted connection to every node in the previous layer (if it is not in the input layer) and every node in the successive layer (if it is not in the output layer). A given node's value is the sum of all previous values, typically with an added bias value. The final output can then be compared with some known label, and the error can be computed using some loss function. Common loss functions include mean squared error and cross-entropy. This error is used to optimize the model through a process called backpropagation, where a gradient descent algorithm is used to adjust the weights and biases to minimize error. Common gradient descent algorithms include stochastic gradient descent and ADAM.

Neural networks can be supervised or unsupervised [10]. Supervised neural networks are most common; they rely on known labels for the training data [35]. Unsupervised networks, on the other hand, adjust parameters without training labels, and rely on alternative methods of training, such as genetic algorithms [59].

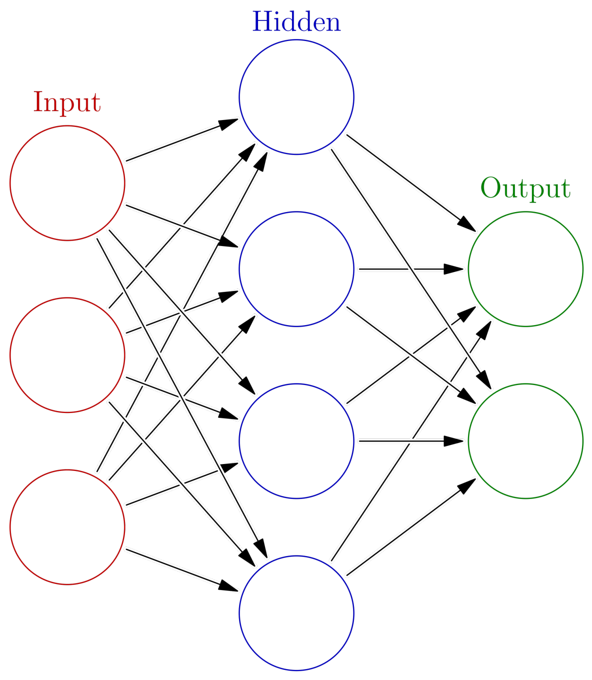


Figure 2.3: Diagram of a simple MLP. More hidden layers and more nodes per layer are common. (Taken from Wikipedia under Creative Commons License 3.0)

Convolutional neural networks (CNNs) significantly outperform prior algorithms at image-related tasks, and new models are superseding prior architectures at a swift rate [24]. The modern theory of CNNs stems from the findings of Hubel and Wiesel, which characterized the function of neurons in the visual cortex interpreting visual information received from the eye.

CNNs are so named because convolutional neural network layers are based on the discrete convolution operation, defined in 2 dimensions below:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

In practice, many machine learning libraries instead implement the cross-correlation function, where the kernel is simply flipped, as shown:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

In contrast with traditional fully-connected neural network layers, convolutional layers have spare interactions, where the kernel is smaller than the input [22]. This results in fewer model parameters and operations, which can significantly improve efficiency of a model. Convolutional layers are capable of detecting features such as edges in an image. Typically, convolutional layers are accompanied by pooling layers, where some operation is leveraged to heavily reduce the dimensionality of the data; for instance, in the case of 2D max pooling, the input is divided into a by b neighborhoods for which the output is simply $\max_{\{a,b\}}(x_{a,b})$. This also means that the network's output is invariant in the face of minor translations of the input, which is beneficial in many image-related tasks.

The modern framework for the CNN was established with the LeNet architecture [36, 37]. Their adoption and improvement accelerated rapidly with a series of key CNNs, particularly those that won the ImageNet competition, including AlexNet [58], followed by deeper and more computationally complex, including VGGNet [62], GoogLeNet [64], and ResNet [26], which leveraged an extremely deep architecture with residual connections.

Image restoration is another key area of application for deep neural networks. Supervised learning with neural networks has been the more common approach; recently, however, unsupervised or self-supervised learning using deep image priors [72] for image restoration tasks has enabled improved performance without the need for training data. In [72], the authors showed that a randomly-initialized neural network can be used as a handcrafted prior with excellent results in standard inverse problems such as denoising, super-resolution, and inpainting. Deep image priors have been adopted across many application domains [41, 21, 54, 76].

Also recently, analysis-by-synthesis techniques have demonstrated impressive capabilities for estimating visual information, particularly for inverse graphics problems [42, 39, 48, 82, 20, 3, 70]. Mildenhall et al. [44] demonstrate how a multilayer perceptron (MLP) coupled with a special layer known as Fourier features [65] can estimate the 5D radiance field of a scene. More recently, the NeRF architecture has been exploited to solve problems like view synthesis, texture completion from impartial 3D data, non-line-of-sight imaging recognition, etc [6, 7, 13, 55, 85].

2.3 Related Work

A variety of methods to mitigate turbulence effects in imaging have been used, including blind deconvolution [9, 19, 88] and optical flow [4, 45, 61]. The primary classical approaches to turbulence removal include spatio-temporal models and lucky frames.

Air and water turbulent images are usually enhanced in different ways. For air turbulence, physics-based approaches use complex turbulence models (e.g., the Kolmogorov model [33, 34]) to simulate the perturbation, and then restore clear images by inverting the models. For water turbulence, classical methods model the distortion as a function of the water surface height or normal by applying Snell's law [67, 84]. Recently, several learning-based methods are separately proposed to remove either air or water turbulence effects.

The problem of turbulence removal can be conceived as the attempt to find some function F that transforms some turbulent image I_t into a clean image I_c without the distortion or blur caused by turbulence. Some models aim to use models that approximate the function based of statistical and/or physics-based properties of turbulence. Zhu and Milanfar provides a spatio-temporal model based on 2D convolution. Methods employing image registration with deformation estimation architecture can also resolve small movements of the camera and temporal variations due to atmospheric refraction [88, 27].

While these models offer a clear mathematically-decipherable method for reversing turbulence in contrast to neural networks, they rely on inaccurate approximations of the turbulence function, and are thus inherently limited in their efficacy.

The lucky frames approach relies on short-exposure frames from a video stream or sequence of images [75, 57, 16]. The image frame is divided into regions. For each region, the best (”lucky”) frame is selected, then all of the regions are fused to form the picture [87]. John and Vorontsov [31] improved this method by searching and fusing the “lucky region”, instead of the whole frame, to restore the scene. Aubailly et al. [2] introduced an automated kernel selection technique to further improve the lucky frames approach.

These methods require enough images (often hundreds) that a sufficiently lucky frame is found for each region. Even with a large enough dataset, this style of approach is outperformed by more modern approaches, particularly those that leverage deep neural networks.

A modern approach to removing atmospheric turbulence relies on deep neural networks [17, 46]. NNs have the ability to approximate highly complex functions without prior knowledge about said functions. These methods often outperform traditional approaches to turbulence performance [5].

However, a key disadvantage of deep neural networks in the restoration of clean images from turbulent images is the lack of available data [73]. Real-world images, both clean and turbulent, of the same scene are difficult to obtain. Therefore, images with simulated turbulence often must be used for supervised models. These models also suffer from poor generalization beyond their training dataset.

Recovering undistorted images from underwater images has been well-studied in computer vision for various applications. Early solutions [14, 38] take the mean/median of a distorted image sequence to approximate the latent distortion-free image, although these methods are limited for large distortions. Like in the case of air turbulence, "lucky region" algorithms have also been proposed using clustering [11, 12], manifold embedding [14], and Fourier-based averaging [78]. The seminal work of [67, 68] presents a model-based tracking method to restore underwater images.

Recent advances in imaging through water distortions have leveraged deep learning for state-of-the-art performance. Li et al. [40] propose a generative adversarial network (GAN) to correct refractive distortions using a single image. The main drawback of Li's method was that it did not leverage the temporal consistent nature of the fluid flow. Thapa et al. [66] propose a two-step dynamic fluid surface reconstruction network to recover the depth and normal maps of the transparent fluid given a short sequence (3 frames) of distorted fluid images.

In a turbulence removal problem similar to the atmospheric examples presented, Xue et al. adapt classical optical flow to estimate small refractive distortions caused by hot air or gas [81].

Many of the methods discussed have artifacts when reconstructing dynamic scenes with large amounts of motion. To counter this, methods have been introduced such as block matching [30], enforcing temporal consistency [47], using reference frames [8], and segmenting static background from moving objects [49, 25, 1]. One promising avenue of direction has been utilizing the physics of turbulence to create accurate forward models for image formation. Mao et al. [43] achieve state-of-the-art performance by utilizing knowledge of atmospheric turbulence to create a physics-constrained prior for optimization.

Chapter 3

DEVELOPMENT OF AN UNSUPERVISED MODEL FOR REMOVING EFFECTS OF ATMOSPHERIC TURBULENCE FROM IMAGES

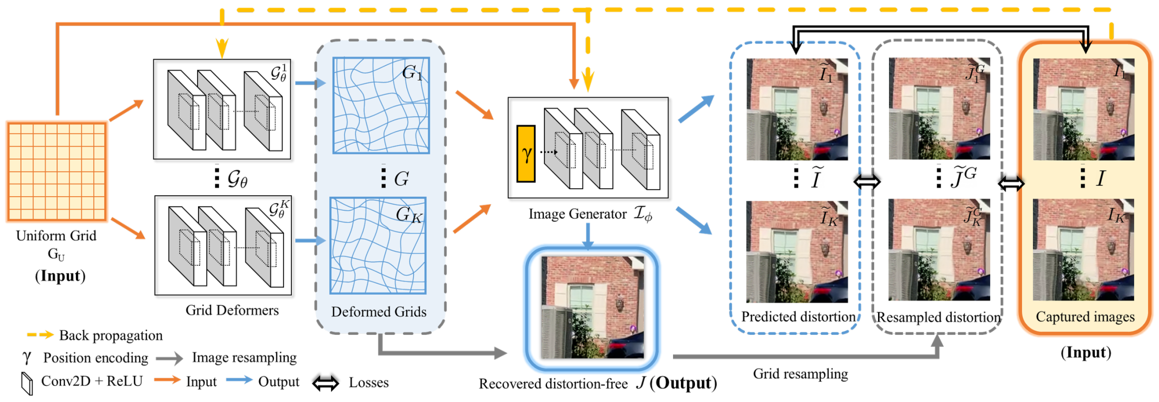


Figure 3.1: The overall architecture of our unsupervised non-rigid image distortion removal network. The network predicts the distortion-free image J, given a sequence of distorted turbulent image $\{I_k|k = 1, 2, ...K\}$ and uniform grid G_U. $\widetilde{I}$ and $\widetilde{J}^G$ are two intermediate results to constrain the optimization procedure. We use the pair-wise differences among I, $\widetilde{I}$, and $\widetilde{J}^G$ as the optimization losses.

Here we present our design for an unsupervised network that is able to remove non-rigid distortions from both air and water turbulent images, as shown in Fig. 3.2. The key idea is to model the non-rigid distortions as a deformable grids. For example, we model the distortion-free image as a straight and uniform grid, and turbulent images with distorted grids. Inspired by recent works on the Neural Radiance Field (NeRF) [44, 65], we generate the distortion-free image using a grid-based rendering network. Our method, therefore, bypasses sophisticated and heterogeneous physical turbulence models and is able to restore images with different types of distortions.

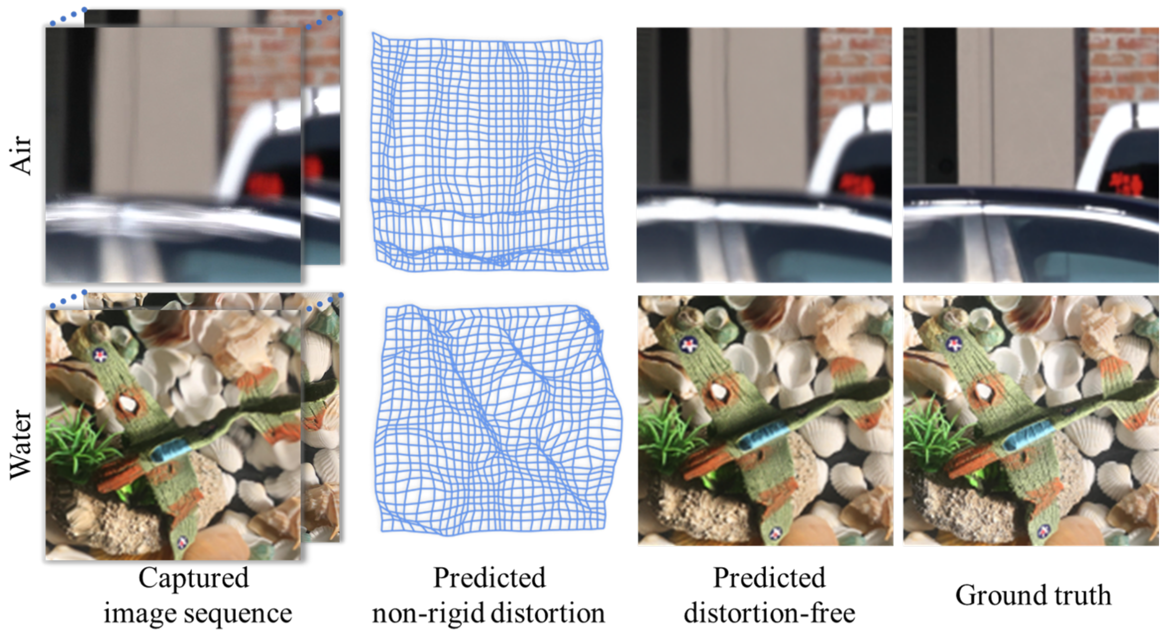


Figure 3.2: We present a novel unsupervised network to estimate the non-rigid distortion and latent distortion-free image when imaging through turbulent media. Our method works for both air (Row One) and water (Row Two) distortions.

The overall structure of our network is illustrated in Fig. 3.1. Our network consists of two main components: a grid deformer $\mathcal{G}$ that estimates the grid deformation and an image generator $\mathcal{I}$ that renders a color image that matches the distortion of an input grid. One critical component in our network is the position encoding operator commonly used in NeRF networks [7, 13, 55, 65, 85]. By incorporating this operator into the image generator, we can simplify our network structure while maintaining fine spatial details in the reconstructed output images.

Our neural network works as an optimizer for generating the distortion-free image by minimizing pairwise differences between the captured input images, the network's predicted distorted images, and resampled distorted images from the distortion-free image. Our network optimizes its parameters based on specific inputs without annotation and does not need to be trained on a labeled dataset. Specifically, our network is optimized in two steps: we first initialize our network parameters by exploiting the locally-centered property [49] of pixel displacement caused by turbulent media; we then iteratively update the estimated distortion-free image J by minimizing our objective function. Empirically, this two-step optimization converges very fast because the initialization step provides a reasonable estimation that largely reduces the search space.

3.1 Non-rigid Distortion Removal Network

Our problem formulation is as follows: we assume a static scene being imaged by a camera with non-rigid distortion being induced by turbulence. Given a sequence of captured non-rigidly distorted images $\{I_k | k = 1, 2, ...K\}$ and a uniform grid G_U, our goal is to recover the latent distortion-free image J as if it was unaffected by the turbulent medium.

Our key idea is to model the non-rigid distortions through grid deformation and reconstruct the distortion-free image J while estimating the distorted image sequence to be consistent with the captured data. To do so, we utilize two sub-networks in our main neural network architecture: a **grid deformer** and an **image generator**. The grid deformer $\mathcal{G}_\theta^k$ is a network to deform a uniform sampled straight grid G_U by estimating the distortion field of the captured frames I_k, and generates a deformed grid $G_k = \mathcal{G}_\theta^k(G_U)$. The image generator is a neural network acting as a parametric function $\widetilde{I} = \mathcal{I}_\phi(G)$ that maps a grid G to an image $\widetilde{I}$. When the grid G_k from the grid deformer is used as input, $\mathcal{I}_\phi$ maps its parameters ϕ to a distorted color image $\widetilde{I}_k$, which is compared to the corresponding image frame I_k. At the same time, feeding a uniform grid G_U to the network $\mathcal{I}_\phi$, we can expect $\mathcal{I}_\phi$ map ϕ to a distortion-free image J, as shown in Fig. 3.1. We also use the predicted distorted grids $\{G_1, ..., G_K\}$

to directly resample J and obtain another set of distorted images $\{\widetilde{J}_1^G, ..., \widetilde{J}_K^G\}$ as intermediate results to constrain the optimization procedure.

Novel to our method is its *unsupervised* learning approach, which means that our network does not require ground truth knowledge of the underlying true distortion-free image J_{true}. Instead, given an image scene, our network works as an optimizer that solves for J by minimizing the pair-wise differences among I, $\widetilde{I}$, and $\widetilde{J}^G$. To properly estimate sharp image details in the image generator, we leverage the latest positional encoding technique in [44, 65] to preserve fine-details in our recovered latent image, without the need for extra convolutional layers with many parameters, which we describe in Section 3.1. To improve the convergence of our network, especially important for learning in an unsupervised fashion, we introduce a novel two-step optimization algorithm to constrain our network described in Section 3.1.

Network structure The overall structure of our non-rigid distortion removal network is shown in Fig. 3.1. Our network has two main components: the grid deformer $\mathcal{G}_\theta$ and the image generator $\mathcal{I}_\phi$. Table 3.1 provides detailed network architecture of the two subnets. In the tables, Conv, BN, ReLU refer to convolution layers, batch normalization and Rectified Linear Unit; γ refers to the GRFF position encoding component.

Grid deformer $\mathcal{G}_\theta$ takes a uniform grid $G_U \in \mathbb{R}^{2\times H\times W}$ as input, where W and H are the sampling number along $x-$ and $y-$axis, and outputs a deformed grid $G_k \in \mathbb{R}^{2\times H\times W}$ corresponding to the distortion field of the distorted image $I_k \in \mathbb{R}^{3\times H\times W}$, i.e., $G_k = \mathcal{G}_\theta^k(G_U)$, where θ is the set of trainable network parameters. $\mathcal{G}_\theta$ comprises four convolution layers, each has 256 channels and ReLU rectifier. To meet the range constraint for G_k, a tangent hyperbolic function is applied to the output layer.

	Input	Filters	Output Shape
$\mathcal{G}_\theta$	Input		$128 \times 128 \times 2$
	Conv, ReLU, BN	$256@1 \times 1$	$128 \times 128 \times 256$
	Conv, ReLU	$256@1 \times 1$	$128 \times 128 \times 256$
	Conv, ReLU	$256@1 \times 1$	$128 \times 128 \times 256$
	Conv, Tanh	$2@1 \times 1$	$128 \times 128 \times 2$
$\mathcal{I}_\phi$	Input		$128 \times 128 \times 2$
	γ (GRFF)		$128 \times 128 \times 256$
	Conv, ReLU, BN	$256@1 \times 1$	$128 \times 128 \times 256$
	Conv, ReLU	$256@1 \times 1$	$128 \times 128 \times 256$
	Conv, ReLU	$256@1 \times 1$	$128 \times 128 \times 256$
	Conv, Sigmoid	$3@1 \times 1$	$128 \times 128 \times 3$

Table 3.1: Detailed architecture of grid deformer $\mathcal{G}_\theta$ and the image generator $\mathcal{I}_\phi$. Here, ReLU and BN stands for Rectified Linear Unit and Batch Normalization, respectively. We use the input image size $128 \times 128 \times 3$ as an illustrative example.

Note that, we train a separate $\mathcal{G}_\theta^k$ for each I_k for two reasons. First, the turbulence field, especially for the air turbulence, is random and has less temporal consistency when the image sequence or video is captured under a standard frame rate, i.e. 30 fps [60, 40, 67, 50]. Using a single network to predict all these random distortion fields is challenging without empirical guidance from ground truth labels and strong temporal consistency constraints. Secondly, the network structure of $\mathcal{G}_\theta$ is simple and has few parameters, and thus we can jointly optimize $\{\mathcal{G}_\theta^k | k = 1, \dots K\}$ with low memory consumption for GPU implementation. Please find a more detailed discussion in Section 3.2.

Image generator $\mathcal{I}_\phi$ renders a color image $\widetilde{I} \in \mathbb{R}^{3 \times H \times W}$ when given a grid input

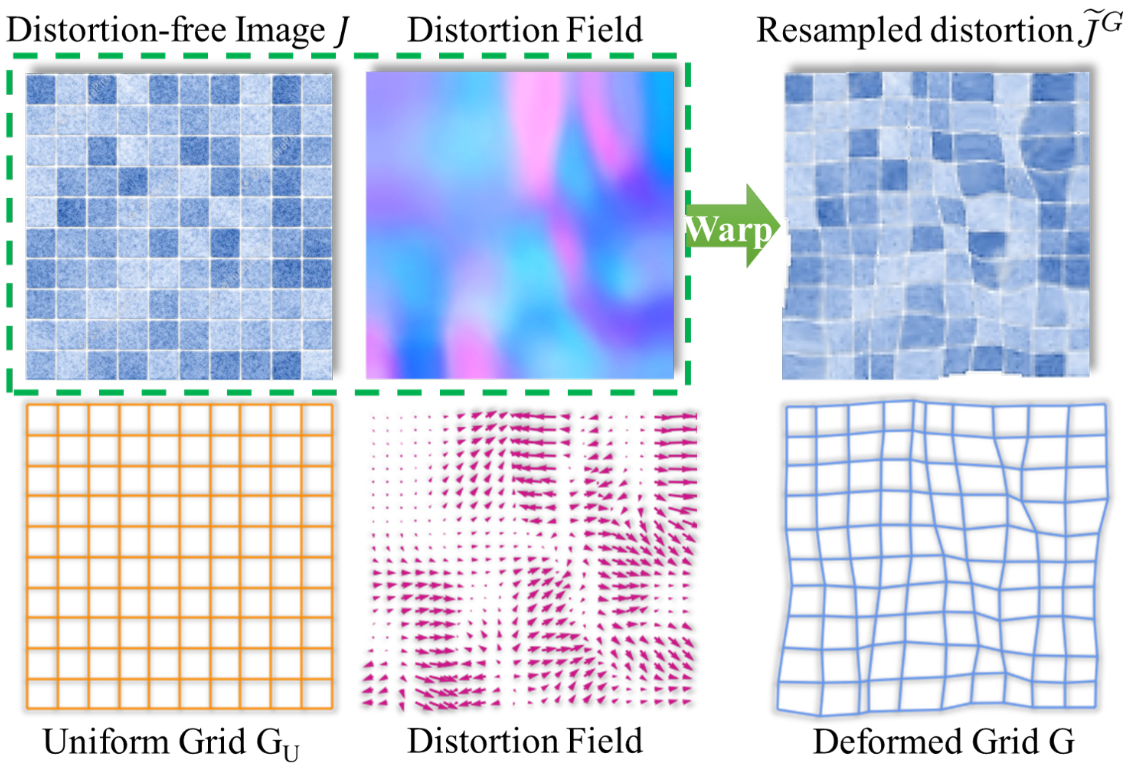

Distortion-free Image J
Distortion Field
Resampled distortion $\widetilde{J}^G$
Warp
Uniform Grid $\mathrm{G_U}$
Distortion Field
Deformed Grid G

Figure 3.3: Distorted image generation via grid deformation.

$G \in \{G_1, \dots G_k, G_U\}$: $\widetilde{I} = \mathcal{I}_\phi(G)$. If the input grid is a deformed grid G_k, $\mathcal{I}_\phi$ returns an image $\widetilde{I}_k$ that matches the distortion of G_k. If the input grid is a uniform grid G_U, we consider the output as a distortion-free image $J \in \mathbb{R}^{3\times H\times W}$. $\mathcal{I}_\phi$ share a similar network architecture with $\mathcal{G}_\theta$. Since the output of $\mathcal{I}_\phi$ is a color image, we apply a nonlinear Sigmoid activation function to the output layer. Please find more details about the structure of $\mathcal{G}_\theta$ and $\mathcal{I}_\phi$ in our supplementary material.

Position encoding via Fourier features As pointed out by [52], networks which directly map xy coordinates to values typically are biased to learn lower frequency functions. To preserve high frequency content in the image, a good solution is to map the grid inputs to a higher dimensional space using high frequency functions before passing them to the network [65, 44]. In our work, we utilize Gaussian random Fourier features (GRFF) to transform the input grid to its high frequency Fourier feature domain before passing it to the image generator $\mathcal{I}_\phi$. Let $\mathbf{v} = (x, y)$ be a coordinate from the input grid. Its GRFF is computed as $\gamma(\mathbf{v}) = [\cos(2\pi\kappa\mathbf{B}\mathbf{v}), \sin(2\pi\kappa_{\mathcal{I}}\mathbf{B}\mathbf{v})]$, where cos and sin are performed element-wise, $\kappa_{\mathcal{I}}$ is a bandwidth-related scale factor, and $\mathbf{B} \in \mathbb{R}^{128\times 2}$ is randomly sampled from a Gaussian distribution $\mathcal{N}(0, 1)$. Thus, the input grid $G \in \mathbb{R}^{2\times H\times W}$ will be mapped into Fourier Feature space $\gamma(\mathbf{v}) \in \mathbb{R}^{256\times H\times W}$.

It is worth noting that the choice of $\kappa_{\mathcal{I}}$ in the image generator is pertinent to our network's performance. In general, large $\kappa_{\mathcal{I}}$ tends to have the network converge fast and very likely to end up at a local minimum. Here we empirically pick $\kappa_{\mathcal{I}} = 8$. We discuss the effect of GRFF in an ablative study in Section 3.2.

Two-step network optimization As our network is unsupervised, it is highly non-convex and has enormous parameter search space. By exploiting redundant information within the deformed image sequence, we propose a two-step network optimization strategy to train a CNN at test time for a given sequence. We first initialize the parameters of $\mathcal{G}_\theta$ and $\mathcal{I}_\phi$ so that they are constrained under properties of non-rigid distortion through turbulent media. Next, we iteratively refine the initialized networks and update the estimated underlying distortion-free image using the captured input distorted images as references.

Parameter initialization. To avoid being trapped in potential saddle points and to allow faster convergence speed, we initialize the network parameters θ and ϕ by exploiting a physical property of pixel displacement caused by turbulent media: *the non-rigid distortions induced by a turbulent medium are generally locally centered* [49]. The distorted images therefore still preserve a large amount of low-frequency image structures. By extrapolating the similarities among the distorted images, we are able to remove a certain amount of non-rigid distortions and obtain a reasonable initial estimation of the distortion-free image.

The grid deformer is initialized by constraining its output to be close to the uniform grid. In this way, we can limit the grid deformation within a certain range and also preserve the order of pixels. We initialize the image generator by constraining its output to have a similar appearance as the input sequence. Specifically, we feed the uniform grid G_U to the image generator. We then compare the output image $J = \mathcal{I}_\phi(\gamma(G_U))$ with all images in $\{I_k\}$, and minimize the sum of per-pixel color differences.

We formulate the initialization procedure as:

$$\min_{\theta,\phi} \sum_k |\mathcal{G}_\theta^k(G_U) - G_U| + |\mathcal{I}_\phi(\gamma(G_U)) - I_k|, \tag{3.1}$$

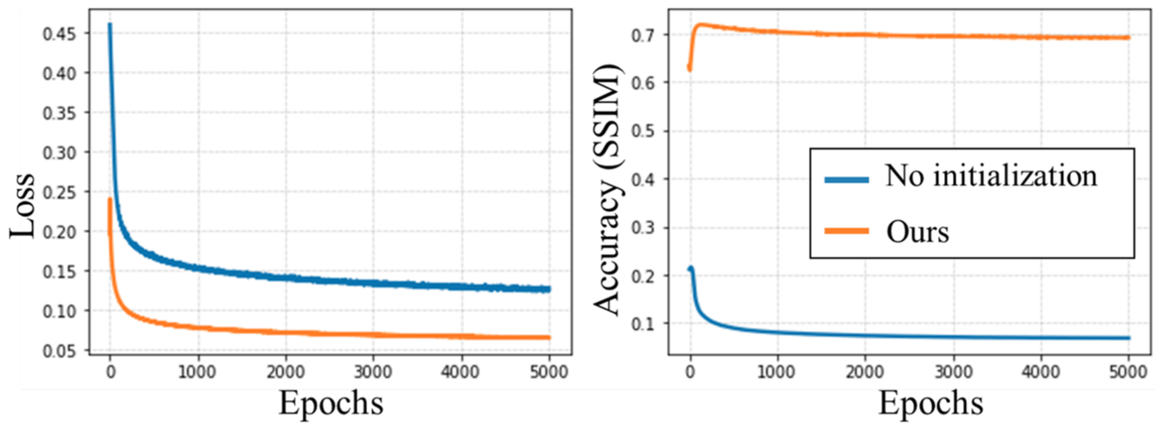
where $|\cdot|$ represents the absolute differences (i.e., the L_1 loss). Notice that we use the L_1 loss for all loss functions as it tends to be less affected by outliers. We run the optimization for a few hundreds of epochs, and use the resulting parameters θ' and ϕ' as the initialized weights.

As illustrated in Fig. 3.4, removing the initialization step will lead the network to converge to a wrong local minimum and fails to predict a reasonable J. In addition, our initialization produces a sharper image that is closer to the latent distortion-free image in color space than simply averaging the images together. This is because taking the average will result in the centroid of the images in RGB color space, and will be blurry since turbulence is time-varying. We discuss more in Section 3.2.

Iterative refinement. After our initialization step, we set out to learn the underlying distortion-free image through the following optimization model:

$$\min_{\theta,\phi} \sum_k |\widetilde{I}_k - I_k| + R(I_k), s.t.\ \theta^0 = \theta',\ \phi^0 = \phi', \tag{3.2}$$

where $\widetilde{I}_k = \mathcal{I}_\phi(\gamma(G_k))$ is the estimated distorted image, $G_k = \mathcal{G}_\theta^k(G_U)$ is the deformed grid, $R(I_k)$ is a regularizer, θ^0 and ϕ^0 are the initial weights of the network. We use



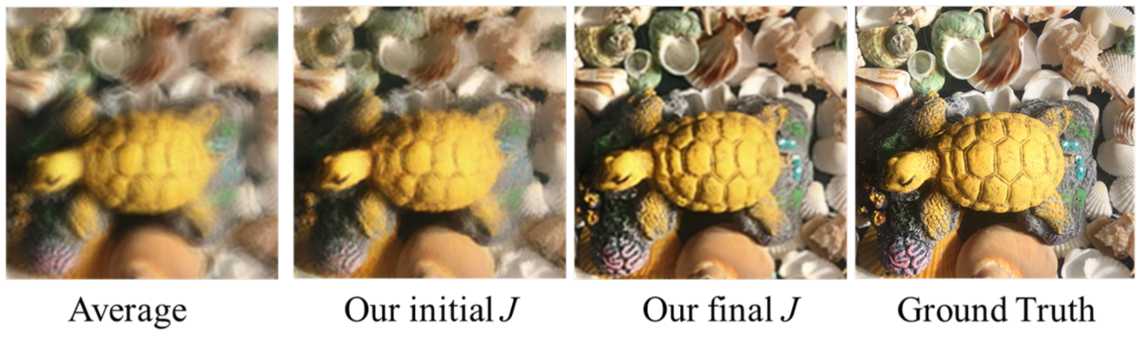


Figure 3.4: The loss and accuracy comparison with and without the initialization step (Top row). Our initialization algorithm improves our prediction performance significantly and can initialize sharper distortion-free images than the simple averaging (Second row).

$R(I_k)$ to strengthen the interconnection between the predicted distortion-free image $J = \mathcal{I}_\phi(\gamma(G_U))$ and deformed grids $\{G_k\}$:

$$R(I_k) = |\widetilde{J}_k^G - I_k| + |\widetilde{J}_k^G - \widetilde{I}_k|, \tag{3.3}$$

where $\widetilde{J}_k^G$ is a resampled distorted image by grid sampling the deformed grid G_k on the recovered latent image J, as shown in Fig. 3.3. We iteratively update the J using Eqn. 3.2 until networks converge.

3.2 Experiments

In this section, we first compare our approach to a set of state-of-the-art methods from the literature on the task of image restoration for both air and fluid turbulence. Then, we present our experimental results to validate our neural network architecture and optimization algorithm. We demonstrate that our method not only outperforms the unsupervised approaches, but even edges out other supervised algorithms that, in contrast to ours, have access to a large amount of synthetic turbulent data using sophisticated physics-based simulators at training time. For quantitative evaluation, we employ the most common metrics for image restoration, i.e., the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM).

Experimental setup Our network was implemented in Pytorch [51] with a desktop computer equipped with two NVIDIA GTX 1080 GPUs. Unless specially stated, the experiments follow the same setting: We use the Adam optimizer and set the learning rate as 10^{-4} for both $\mathcal{G}_\theta$ and $\mathcal{I}_\phi$. We use 1,000 iterations for parameter initialization, and in the iterative refinement stage, our network converges within 1,000 epochs, as shown in Fig. 3.4. We empirically pick $\kappa_{\mathcal{I}} = 8$ as the bandwidth-related factor of the Fourier feature mapping operator for all experiments.

The overall network to handle 10 input frames has around 1.53 million trainable parameters, which include 1.33 million (M) total for the grid deformers (one for each frame) and 0.2 M for the image generator. Compared to a contemporary GAN to restore imaging through water turbulence with about 50 million parameters [40], our network restores comparable high-frequency details in the predicted image with less memory footprint.

We use the physics-based simulation software presented in [60] to render images affected by the air turbulence. Fig. 3.5 illustrates the simulation setup and the respective parameters used in the simulator. Numerical values of the parameters are given in Table 3.2.

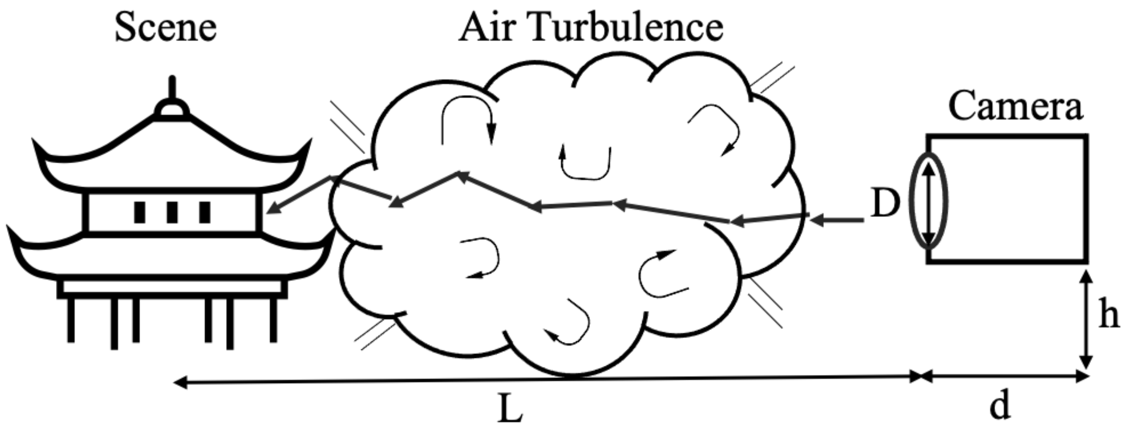


Figure 3.5: Air turbulence simulation setup. We simulate the distorted image of the scene through the turbulent air. Here L is the path length from camera to scene; d is the camera's focal length; h is the camera height; and D is the camera's aperture size.

The simulator uses the refractive image constant (C_n^2) to control the strength of the air turbulence. Stronger turbulence results in more distorted images. In our simulation, we use three levels of C_n^2 to render images under weak, medium, and strong air turbulence. Fig. 3.6 shows exemplary images of the three turbulent strengths.

Evaluation on air turbulence For the air turbulence, we compare with the following state-of-the-art methods: Clear [1], [49], [83], [17], and [43]. [1, 83, 43] are physics-based approaches that use complex turbulence models. [17] is a supervised method trained on a large semi-synthetic turbulence dataset.

We compare the image restoration performance on both real and synthetic datasets. For synthetic experiments, we synthesize turbulent image sequences with different

Parameter		Value
Path length L		2km
Height h		4m
Aperture Diameter D		0.08m
Focal Length d		0.3m
Wavelength λ		550nm
Turbulence	Weak	1×10^{-14} m$^{-2/3}$
Strength	Medium	1×10^{-13} m$^{-2/3}$
C_n^2	Strong	1×10^{-12} m$^{-2/3}$

Table 3.2: Air turbulence simulation parameters.

turbulence strengths. We use the turbulence strength parameter $C_n^2 = 1 \times 10^{-14}$ for the weak turbulence; $C_n^2 = 1 \times 10^{-13}$ for the medium; and $C_n^2 = 1 \times 10^{-12}$ for the strong. The quantitative comparison results with respect to various turbulence levels are reported in Table 3.3. We can see that our method is robust for the strong turbulence.

As for the real data, we compare on two types of air-turbulence phenomena: hot-air turbulence and long-range atmospheric turbulence. For the former, we capture our data by using a gas stove to heat the air. We use a cellphone camera to capture 5 scenes around 50 meters away from the heat source. For the latter, we use data from two sources: (1) the widely adopted *Chimney* and *Building* sequences [28] and (2) our own turbulent images captured using a Nikon Coolpix P1000 camera. We mount the camera on a tripod to capture 1080p videos at 30 fps of 5 scenes at around 1-3 miles away with 125× optical zoom.

We show the comparisons with the state-of-the-arts on *Chimney* and *Building*

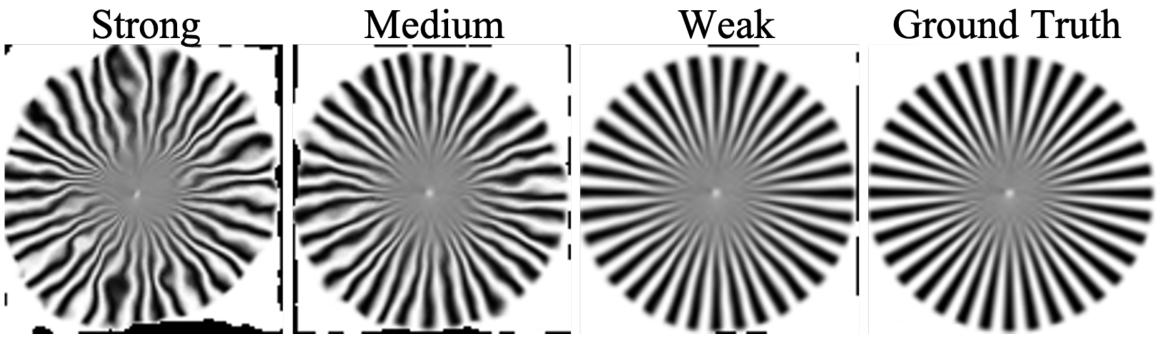


Figure 3.6: Exemplary images of the three turbulent strength levels (weak, medium, and strong) in comparison with the distortion-free image (taken from Open Turbulent Image Set (OTIS) [19]).

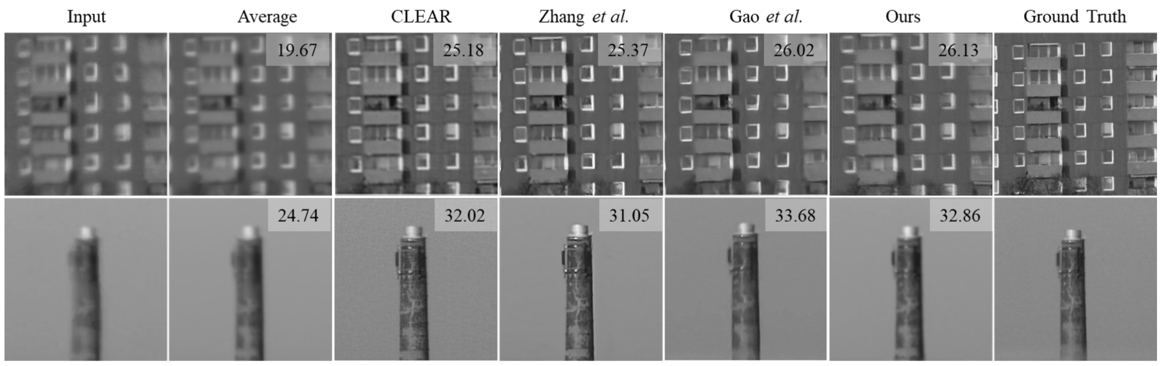


Figure 3.7: Comparisons on the *Building* and the *Chimney*. We also report the PSNR measurement for each restored image. It's worth noting that all methods take the full sequence (100 frames), while our method only takes 10 randomly picked frames.

in Fig. 3.7. As we don't have access to the codes of several methods [1, 83, 17], we directly take the images and reported PSNR from their original papers. It is important to note that most of these algorithms take a longer input sequence (≥ 100 frames) and has deblurring component to produce sharper images. In contrast, our network only needs 10 input frames to make a reliable prediction. But as our network

Strength	Metrics	Average	Our init.	[1]	Ours
Weak	PSNR↑	25.10	**25.20**	18.31	24.29
	SSIM↑	0.941	0.95	0.856	**0.984**
Medium	PSNR↑	19.48	19.85	14.09	**20.70**
	SSIM↑	0.774	0.804	0.561	**0.904**
Strong	PSNR↑	17.08	17.12	12.51	**17.40**
	SSIM↑	0.632	0.667	0.433	**0.799**

Table 3.3: Quantitative comparison on air turbulence data with various strengths. We compare with the temporal average frame, our initial J and CLEAR [1].

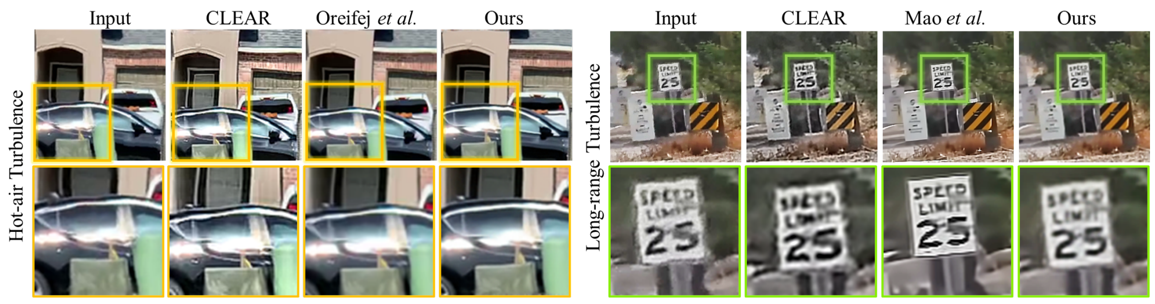


Figure 3.8: Visual comparison results on our real captured hot-air turbulence and long-range atmospheric turbulence images.

focuses more on distortion removal, our output may still suffer from certain amount of blurriness. To help predict sharper images, we can apply a deblurring algorithm on our predicted distortion-free images.

Xu et al. [80] provided a method for motion deblurring, which we applied to the output image of our network. The results presented in Fig. 3.9 demonstrate a qualitative improvement in the output and a substantial reduction in blur.

We show the qualitative comparison on our real captured data in Fig. 3.8. Here

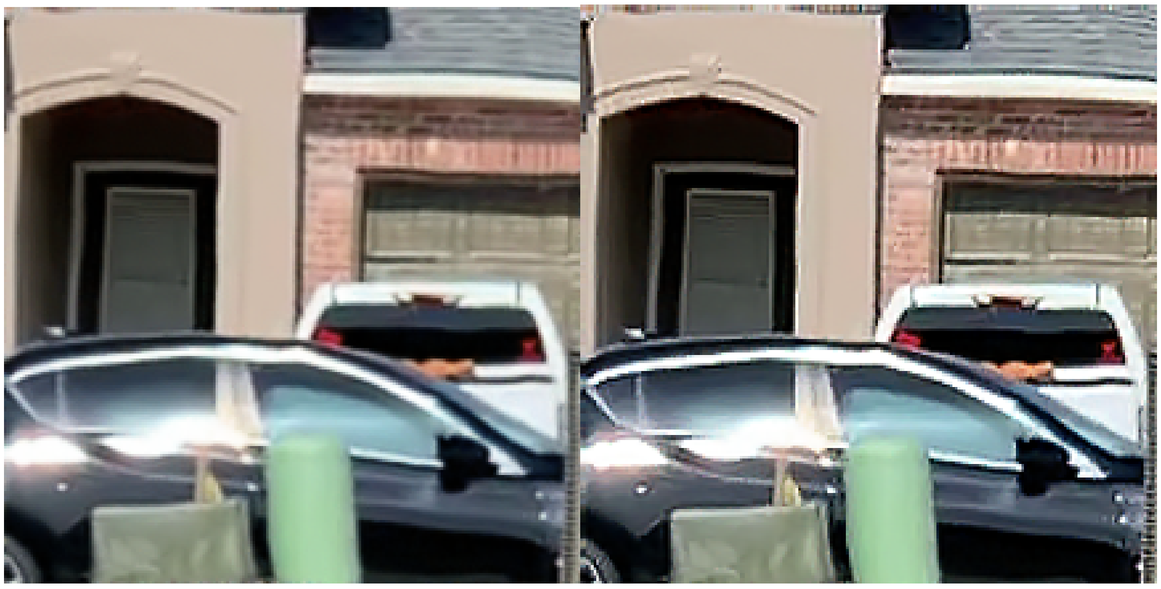

(a) Original (b) Deblurred with [80]

Figure 3.9: Effect of sharpening approach from [80] on the output of our network.

we only compare to the methods that we have access to their codes or the authors have provided us their results.

Evaluation on water turbulence For the water turbulence, we compare our methods with the following state-of-the-arts: [67] and [50] are physics-based methods. [40] is a learning-based method. All provided the source codes.

We perform experiments on two water turbulent image datasets: [66] and [40]. Thapa *et al.* proposed a synthetic dataset providing both the distorted image sequences and the ground truth pattern. The images are simulated using a physics-based ray tracer with different types of waves. [40] is a real captured dataset. It poses challenges such as illumination change and shadows. The distortions are also more drastic. We show the visual comparison results in Fig. 3.10. We can see that our method outperforms all the states-of-the-arts. To further validate the robustness,

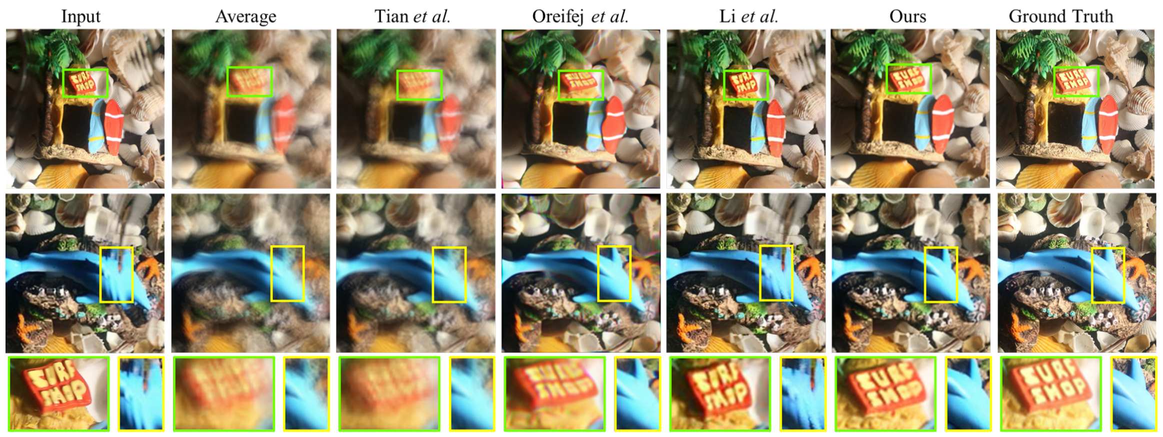


Figure 3.10: Visual comparisons on real water turbulence images provided by [40], which proposed a supervised GAN model to restore water turbulence.

we created three synthetic sequences of water turbulence images, each contains 10 frames, caused by different types of waves using the physics-based ray tracer provided by [66]. The ocean waves are the most challenging, as they are more random and have more high-frequency turbulence components. As shown in Table 3.4, our method ranks higher on the Ripple and Ocean waves. Although [40] achieves higher PSNR/SSIM scores on the Gaussian wave, their results appear blurrier than ours (see visual comparisons in Fig. 3.14). Further, [40] requires training on ∼320K images.

Ablation Studies We conducted a set of ablation studies to validate various design choices in our network architecture. For all these studies, we tested image restoration through simulated air turbulence, as the resulting non-rigid distortions are more random than water turbulence in general. We utilize a physics-based atmospheric turbulence simulator for 2D images [60] to generate 100 different turbulence fields with controllable turbulence strength C_n^2 that are applied to a clear image to generate distorted image sequences.

Network structures of $\mathcal{G}_\theta$. For a fair comparison of the capability of different

Types	Metrics	[67]	[50]	[40]	Ours
Ripple	PSNR↑	20.40	21.24	20.70	**23.63**
	SSIM↑	0.878	0.902	0.882	**0.970**
Ocean	PSNR↑	20.93	21.13	21.32	**22.32**
	SSIM↑	0.891	0.901	0.833	**0.964**
Gaussian	PSNR↑	17.61	17.40	**18.67**	17.50
	SSIM↑	0.787	0.789	**0.833**	0.818

Table 3.4: Quantitative comparison on different types of water turbulence. We compare our result with [67], [50], and [40].

structures in encoding the deformed grid, we replace $\mathcal{G}_\theta^k$ with several different CNNs, as shown in Table 3.5. Specifically, Con_2, Con_4 and Con_6 are CNN structure with 2, 4, 6 convolutional layers respectively. As we have 10 frames in the input sequence, the total number of parameters is equal to 10 $\times$ the size of each $\mathcal{G}_\theta^k$. We also compare with the architecture that simply use a deep autoencoder CNN (DAE) with skip connections [54] to predict 10 deformed grids $\{G_k\}$ at once. We demonstrate that the proposed structure (Con_4) is superior to other networks w.r.t. the restoration ability with fewer trainable parameters.

Number of input images. One critical design consideration for our network is the number of input images needed to generate a distortion-free image. There is a trade-off between the speed of the network in restoring images versus the visual fidelity. In Table 3.6, we show the average PSNR/SSIM and total run time (2,000 epochs) for 2, 5, 10, 15 and 20 frames. A visual comparison is shown in Fig. 3.12. Increasing the input number does benefit our restoration task, but we sacrifice time efficiency

$\mathcal{G}_\theta$	10 subnets			1 network
	$Conv_2$	$Conv_4$	$Conv_6$	DAE
Total params	0.02M	1.33M	2.65M	2.35M
PSNR↑	19.23	**20.48**	20.06	16.83
SSIM↑	0.775	**0.790**	0.742	0.467

Table 3.5: Comparison of the Performance on the Restoration Ability among Different Network Structures of $\mathcal{G}_\theta$

# of inputs	2	5	10	15	20
PSNR↑	17.55	18.50	20.50	**21.43**	21.13
SSIM↑	0.556	0.666	0.793	0.827	**0.830**
Time	65s	143s	265s	403s	499s

Table 3.6: Average PSNR, SSIM, and run time (2,000 epochs) comparison on taking different numbers of input images.

in order to do so. Since there are diminishing returns to the image quality of our predicted sharp images after 10 input frames, this number is chosen as the default input number throughout the following experiments.

Effect of position encoding. The Gaussian random Fourier features (GRFFs) encodes the input grid into a higher dimensional space, enabling our image generator to approximate real high-frequency sharp images. We compare our full network with the one that removes the GRFF in the image generator and simply takes $\{G_k\}$ as input. As shown in Fig. 3.11, with Fourier feature mapping operators in $\mathcal{I}$, we have about 30.9% improvement in SSIM and 13.9% improvement in PSNR of the recovered latent

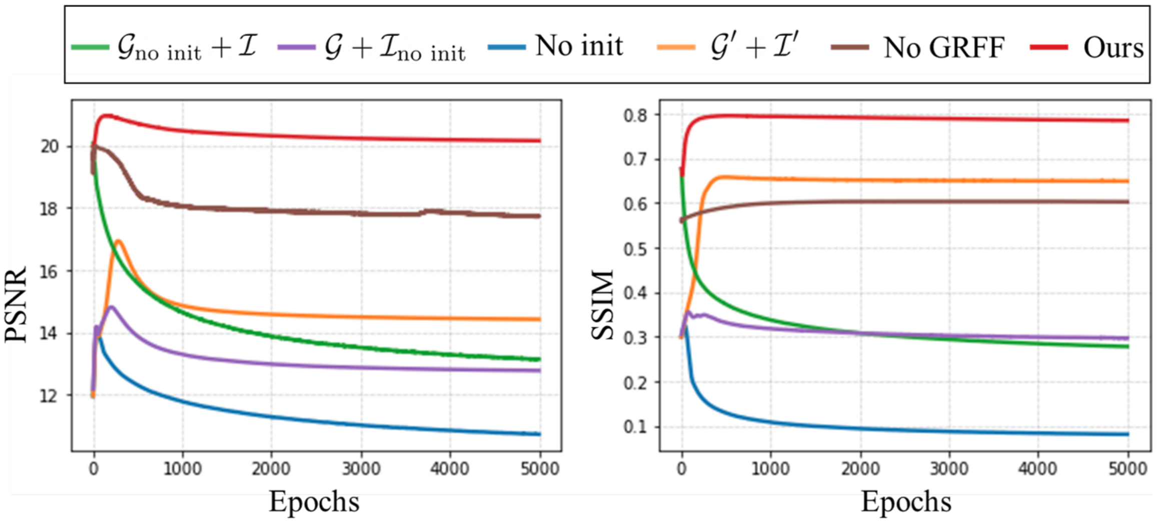


Figure 3.11: Ablation study on different variants of our proposed network. We show the PSNR and SSIM vs. the number of iteration curves for comparison.

images, compared with the network variant without GRFF (No GRFF). They also help increase the convergence speed. We show the impact of the bandwidth-related scale factor $\kappa_{\mathcal{I}}$ in Fig. 3.12.

Effect of initialization step. To evaluate the effects of the network initialization, we created four variants of the network for comparison: 1) $\mathcal{G}_{\text{no init}} + \mathcal{I}$, that removes the initialization step of grid deformer $\mathcal{G}$; 2) $\mathcal{G}+\mathcal{I}_{\text{no init}}$, that removes the initialization step of image generator $\mathcal{I}$; 3) *No init*, that has no initialization step at all; and 4) $\mathcal{G}' + \mathcal{I}'$, that adds the initialization losses to the iterative refinement step. As shown in Fig. 3.11, taking out the initialization step from either the $\mathcal{G}_\theta$ and $\mathcal{I}_\phi$, the overall network has subpar optimization performance and fails to predict a reasonably sharp image. However, simply adding the initialization losses to the main optimization loop can degrade our restoration performance, as these losses can lead the network to converge to some local minimum, as discussed in Section. 3.1.

Effect of number of input images Here Fig. 3.12 shows the visual comparison results of different number of input images. We can see that the qualitative improvement of the predicted distortion-free image becomes margin when the number of input images is greater than 10. We therefore take 10 input images as the default setting to balance performance and efficiency.

Effect of GRFF parameters We evaluate the impact of the bandwidth-related scale factor $\kappa_{\mathcal{I}}$ for Fourier feature mapping. We show the quantitative comparison results (average PSNR/SSIM) in Table 3.7, and qualitative comparison results on distortion-free image and distortion field prediction in Fig. 3.12.

$\kappa_{\mathcal{I}}$	0.1	1	8	10	50	100
PSNR	19.38	19.29	20.28	20.01	16.24	13.89
SSIM	0.627	0.800	0.796	0.754	0.373	0.372

Table 3.7: Quantitative comparison on varied Fourier feature mapping parameters $\kappa_{\mathcal{I}}$. Red and Blue refer to the top and second best performance respectively.

3.3 More Results on Synthetic and Real Data

Air turbulence results We show additional visual results on simulated air turbulence in Fig. 3.13. We compare on simulated air turbulence of three strength levels: weak, medium and strong. We compare with the state-of-the-art method CLEAR [1], whose source code is available. We also compare with the average image of the entire input sequence, as well as our initialization result.

We show more results on the real captured hot-air turbulence scenes in the supplementary video. The videos are filmed by imaging through the hot air generated

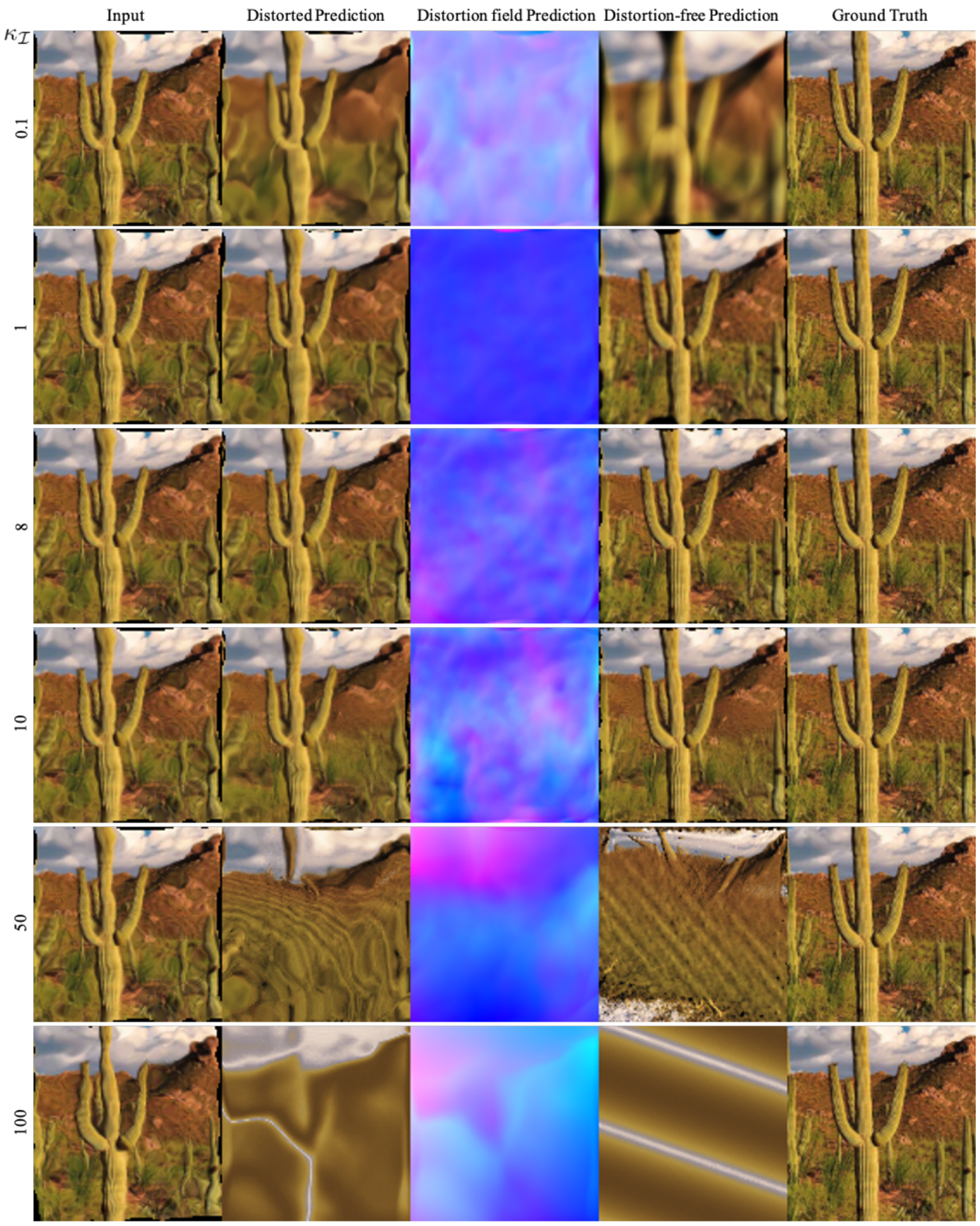


Figure 3.12: Qualitative comparison on varied values for $\kappa_{\mathcal{I}}$ and different numbers of input images (1, 5, 10, 15, and 20).

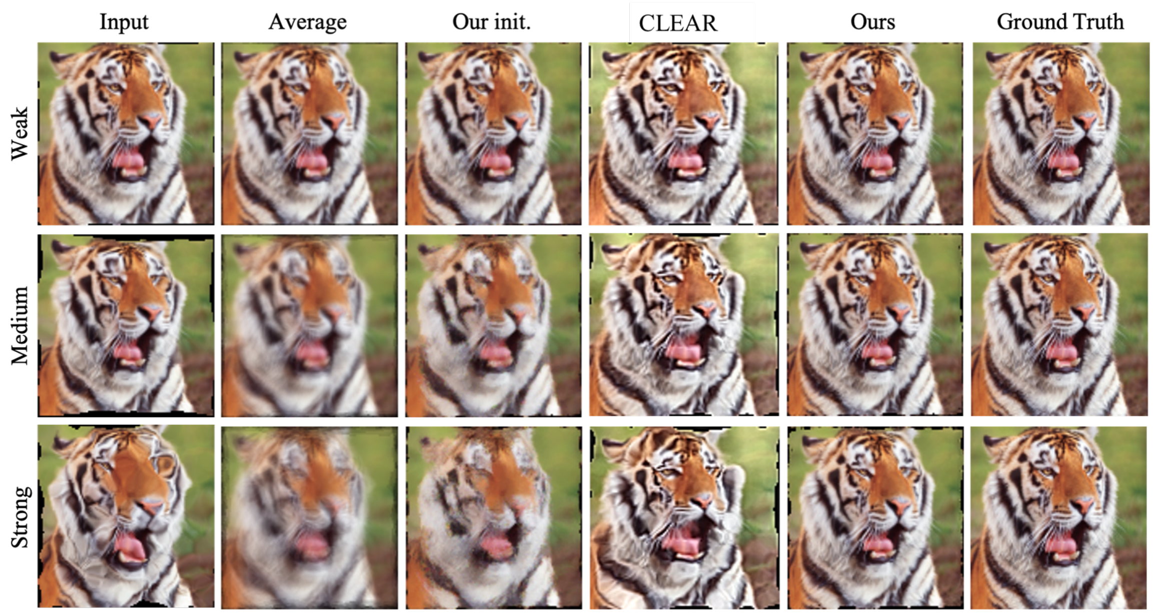


Figure 3.13: Qualitative comparison on simulated air turbulence images with various strengths (weak, medium, and strong).

by a lit gas stove. We include video clips of two scenes. Each scene has 50 frames. As recovering all frames together is computationally expensive, we divide the 50 frames into three batches: 20, 20 and 10. We then feed these three batches into our network to predict the distortion-free image and the distortion fields. In the videos, we show the predicted distortion-free image from the last batch.

Water turbulence results We show additional visual results on simulated water turbulence in Fig. 3.14. We compare on simulated water turbulence of three types: ripple, ocean and Gaussian. We compare with the state-of-the-art methods Tian et al. [67], Oreifej et al. [50], and Li et al. [40].

The real captured turbulent videos are provided by [66]. The videos are captured through a wavy water surface. We choose two different types of waves with different background patterns. The video processing method is similar to the air turbulence

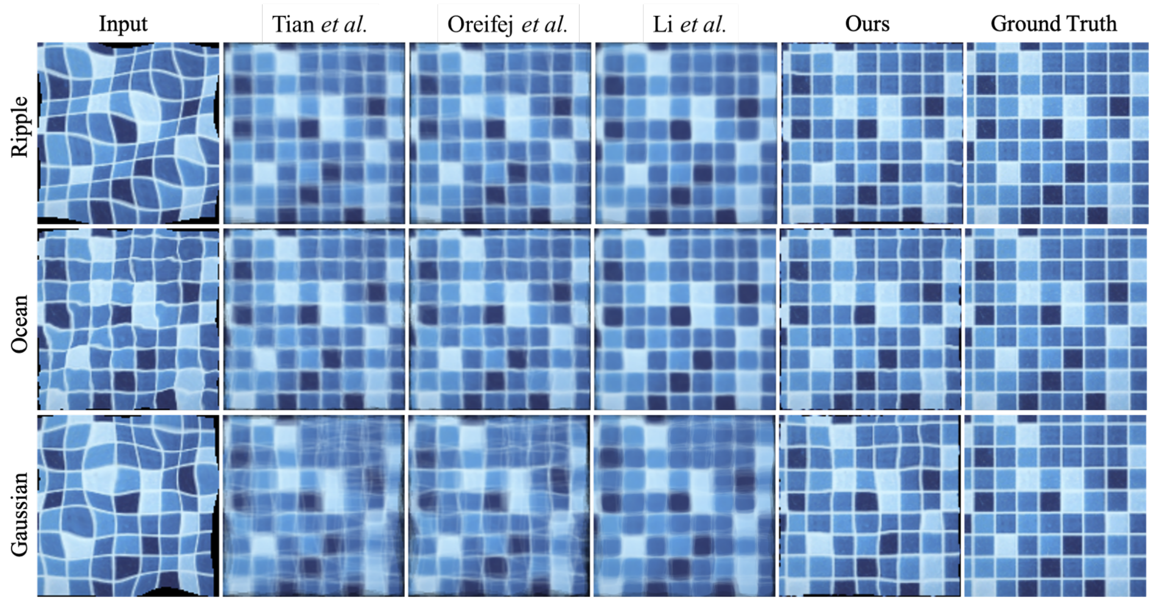


Figure 3.14: Qualitative comparison on simulated water turbulence images with various types of waves (ripple, ocean, and Gaussian).

case. Each scene has 50 frames. We divide the 50 frames into three batches: 20, 20 and 10. We then feed these three batches into our network to predict the distortion-free image and the distortion fields. In the videos, we show the predicted distortion-free image from the last batch.

In order to combat camera jitter, we stabilized the video input to the network using the Adobe Premiere Pro software. The results are shown in Fig. 3.15. The stabilization of the input results in a relatively clear image. In contrast, the unstabilized input video frames resulted in an output that shows artifacts associated with disparate locations of points in different frames. Still, the stabilized output does show smaller artifacts, and does not display the exact same region of the input video as is shown in the input.

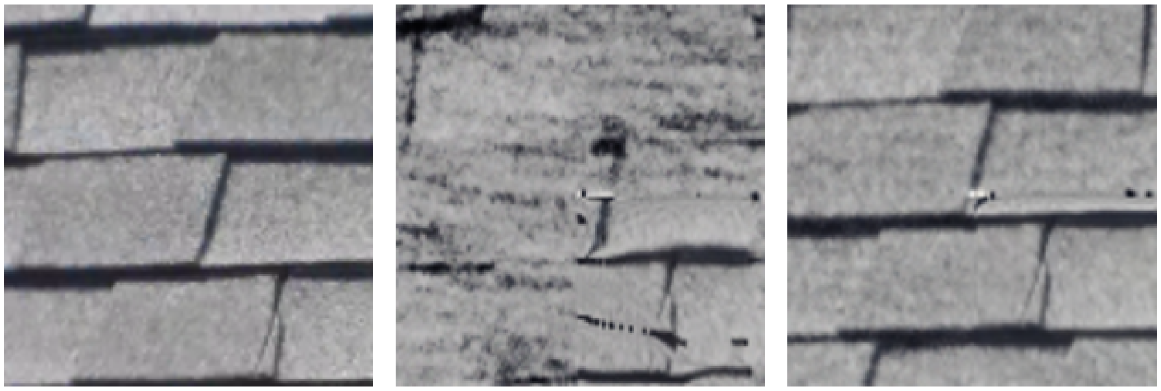

(a) House input (b) Unstabilized Output (c) Stabilized

Figure 3.15: Network results on unstabilized vs. stabilized inputs.

Chapter 4

ESTIMATION OF THE REFRACTIVE INDEX STRUCTURE CONSTANT FROM WARP FIELDS

Knowing the utility of estimation of the refractive index structure constant, C_n^2, we sought to provide a method for C_n^2 estimation that only required an image. We leveraged an additional deep convolutional neural network model alongside a physics-based turbulence simulator to accomplish this goal. The process behind this method and results from experiments for C_n^2 estimation are describe in this chapter.

4.1 Existing C_n^2 Estimation Methods

Few methods exist that leverage deep learning to estimate C_n^2. Those that do use instrument data as inputs for training data, and scintillometer measurements as training labels for supervised learning.

[77] trains on temperature, relative humidity, barometric pressure, potential temperature gradient, and wind shear as inputs into a small MLP with a single hidden layer containing 5 nodes, which outputs an estimated C_n^2 value. [63] presented 3 variations on their own model, and the best-performing trained both with a combination of the typical backpropagation algorithm and an adaptive niche-genetic algorithm. Its inputs were temperature, wind velocity, and relative humidity each at 0.5m and 2m above the ground as well as pressure and snow surface temperature. Similarly to the prior example, they used an MLP with a single hidden layer containing 6 nodes, and had $log(C_n^2)$ as output. 4.1 the model has shows good accuracy with a $log(C_n^2)$ between -13.5 and -14.5, but decreased accuracy for $log(C_n^2)$ values outside this range. [74] proposed a model trained on intensity scintillation images captured

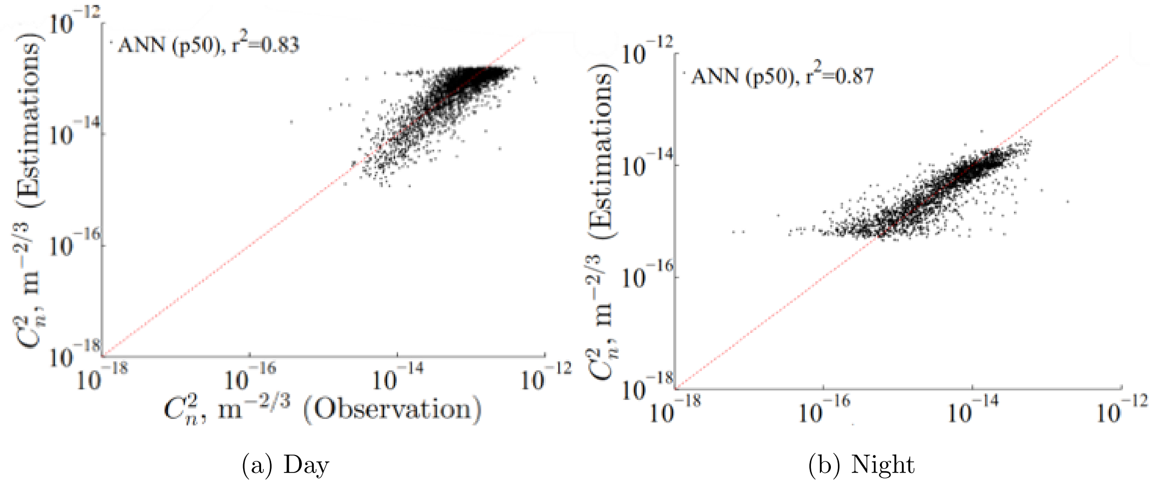


Figure 4.1: Scatterplot comparing real and estimated C_n^2 values for day and night, as presented by Wang and Basu. Reprinted with permission from [77] ©The Optical Society.

alongside (C_n^2) values using a scintillometer setup. The model was a convolutional neural network trained on 3 different datasets each with 9000 or more images over a range of C_n^2. As with the other models, estimation had higher error with lower C_n^2 values.

4.2 Turbulence Simulator

The original simulator [60] is designed to effectively render 2-dimensional image distortions consistent with 3-dimensional turbulence; see Fig. 4.2. The theory proposed in the paper is summarized in a set of 2D operations, which together compose the formula for rendering an image with atmospheric turbulence using a few parameters from the atmosphere and camera. The paper also demonstrates methodology verifying that the distortion statistics of the generated warp fields are consistent with the theory described.

Each pixel's distortion can be described by Fig. 4.3. **o** represents the line of sight

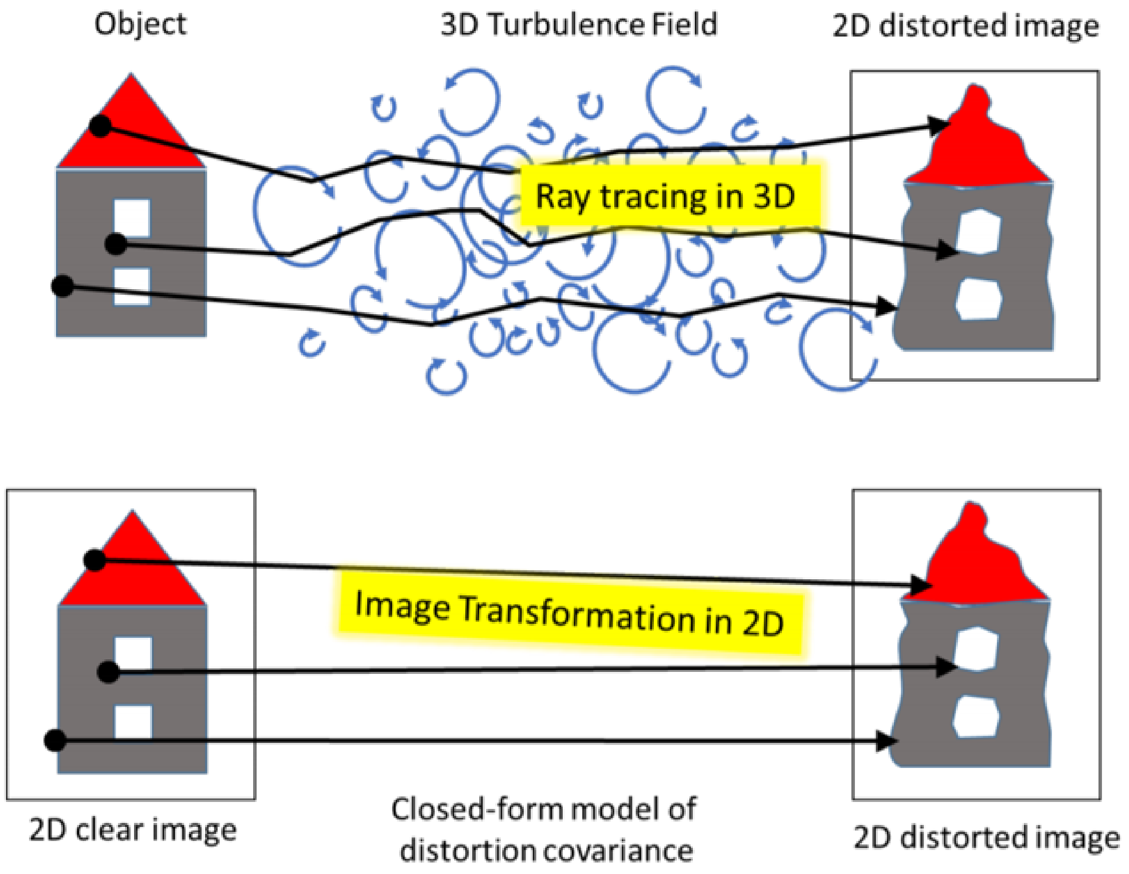


Figure 4.2: Rendering air turbulence in 3D vs. 2d, as shown in [60]. ©2017 IEEE.

to a point o in the scene. The angle of arrival of light from point o is perturbed by the refractions through the turbulent medium, so that the point at which the light arrives on the sensor is changed from $\mathbf{p}$ to $\mathbf{p} + \mathbf{e}(\mathbf{p})$. Here L represents the length of the light's path, D the diameter of the aperture, and f the focal length of the camera.

The recipe presented in [60] is as follows (modified to include equations):

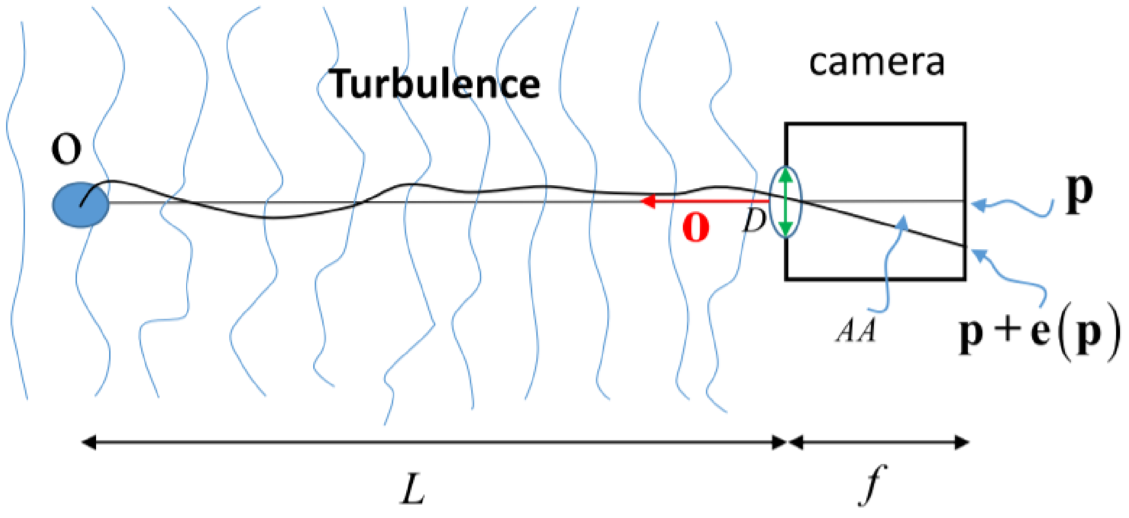


Figure 4.3: Camera diagram demonstrating the change in angle of arrival of light resulting from turbulence, as taken from [60]. ©2017 IEEE.

1) Calculate $\mathbf{C}(\mathbf{v})$ using the following equations:

$$\mathbf{C}(\mathbf{v}) = A(r)\mathbf{I} - B(r)\hat{\mathbf{v}}\hat{\mathbf{v}}^{\intercal}$$
$$A(r) = b_{\perp}\left(\frac{r}{f\theta_D}\right)$$
$$B(r) = b_{\perp}\left(\frac{r}{f\theta_D}\right) - b_{\parallel}\left(\frac{r}{f\theta_D}\right)$$

2) Fourier transform $\mathbf{C}(\mathbf{v})$ to $\tilde{\mathbf{C}}(\boldsymbol{\omega})$.

3) Calculate filter $\tilde{\mathbf{K}}(\boldsymbol{\omega})$ using the following equation:

$$\tilde{\mathbf{K}}(\boldsymbol{\omega}) = \frac{1}{t(\boldsymbol{\omega})}\left[\tilde{\mathbf{C}}(\boldsymbol{\omega}) + s(\boldsymbol{\omega})\mathbf{I}\right]$$

4) Sample a random Gaussian white noise field $\mathbf{z}(\mathbf{p})$.

5) Fourier transform $\mathbf{z}(\mathbf{p})$ to $\tilde{\mathbf{z}}(\boldsymbol{\omega})$.

6) Use $\tilde{\mathbf{K}}(\boldsymbol{\omega})$ in a simple multiplication per ω to obtain $\tilde{\mathbf{e}}(\boldsymbol{\omega})$, as written in the

following equation:

$$\tilde{\mathbf{e}}(\boldsymbol{\omega}) = \int \mathbf{e}(\mathbf{p}) e^{-j\boldsymbol{\omega}^\intercal \mathbf{p}} d\mathbf{p} = \tilde{\mathbf{K}}(\boldsymbol{\omega}) \tilde{\mathbf{z}}(\boldsymbol{\omega})$$

7) Inverse Fourier transform $\tilde{\mathbf{e}}(\boldsymbol{\omega})$, to obtain the distortion field $\mathbf{e}(\mathbf{p})$, of variance 1.

8) Amplify $\mathbf{e}(\mathbf{p})$ so its variance is consistent with the following equation:

$$\sigma^2_{\text{AA}} = \gamma(q) \mathcal{C}_n^2 L D^{-1/3}$$

For further details, including the derivation of these equations, consult the paper. Additionally, an implementation of these equations in available in MATLAB as published by the authors.

4.3 Preliminary Experiments - Differentiable C_n^2 Estimation

We adapted the existing turbulence simulator from [60] into the PyTorch framework within Python 3. Fig. 4.5 demonstrates our adaption accurately reflects the capabilities of the original simulator. By adapting the simulator into PyTorch, we aimed to leverage PyTorch's autograd feature to make the simulator automatically differentiable. The structure of the simulator we implemented is shown in 4.4. Since each of the modules that incorporate C_n^2 into their calculations are differentiable, and they all lead up to the generation of a turbulent image, it is possible to impose a loss between two different turbulent images and backpropagate the resulting error to correct the single adjustable parameter, C_n^2 (see Fig. 4.7). The turbulent image generated by the forward pass with the parameters at the given stage of training is compared with the true turbulent image. This enabled us to estimate C_n^2 if we know all other parameters, including the random tensors used to generate the warp field. With all other values known, the differentiable simulator is capable of consistently converging

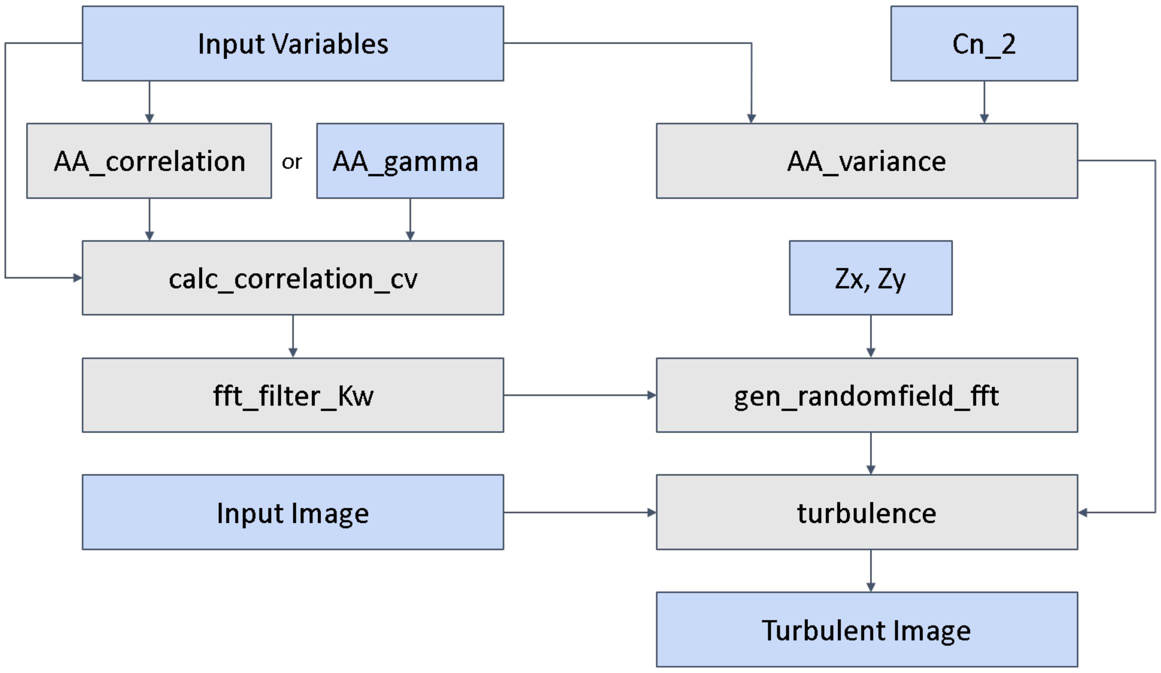


Figure 4.4: Differentiable simulator code structure. Modules are shown in gray; inputs are shown in blue.

on the correct C_n^2 value; this is demonstrated in 4.6. We ran the differentiable simulator with a variety of initializing C_n^2 values and ground truth C_n^2 values, and this consistently held true. This method then functions as an unsupervised estimator of C_n^2.

However, this method used alone has practical limitations. In practice, the white noise vector field components Z_x and Z_y are not known; therefore, they too must be estimated. Since the modules which depend on Z_x and Z_y are also differentiable, it is possible to also backpropagate the error to adjust the individual values in Z_x and Z_y. We attempted a series of experiments to determine whether these white noise fields could be estimated using the warped image. We attempted to fix Z_x and estimate only Z_y, estimate Z_x and Z_y simultaneously but with a fixed known C_n^2 value, and finally estimate Z_x, Z_y, and C_n^2 simultaneously. In order to make convergence

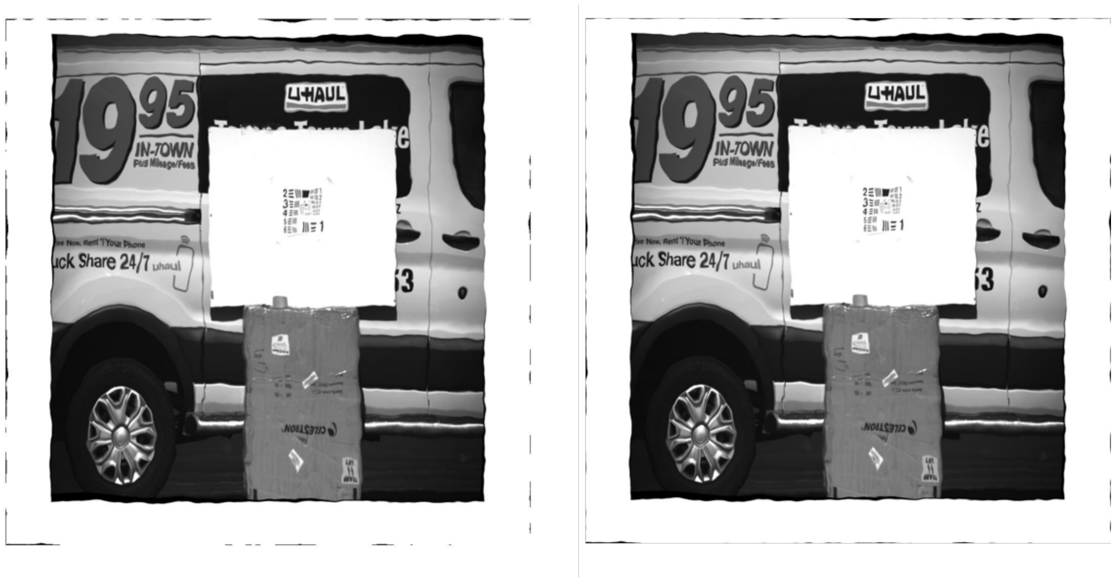
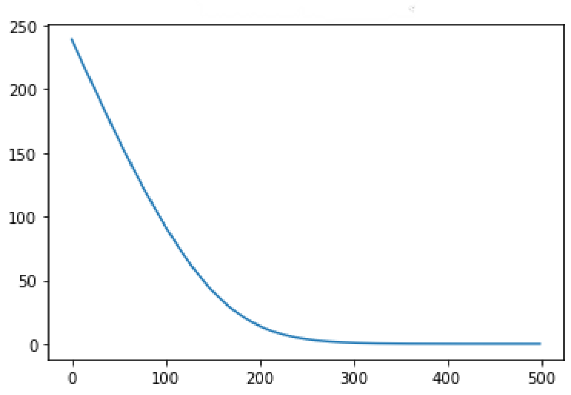
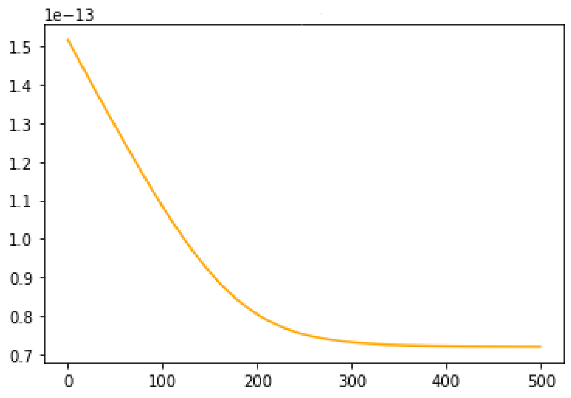


Figure 4.5: Side-by-side comparison of MATLAB (left) and PyTorch (right) turbulence simulator results using identical input image and parameters shows that both simulators produce nearly identical results.



(a) Loss over Epochs



(b) Estimated C_n^2 value over Epochs

Figure 4.6: Loss and estimated C_n^2 over 500 epochs by differentiable simulator given all other inputs are known. The estimator converges to the true value of $C_n^2 = 7.203e{-}15$.

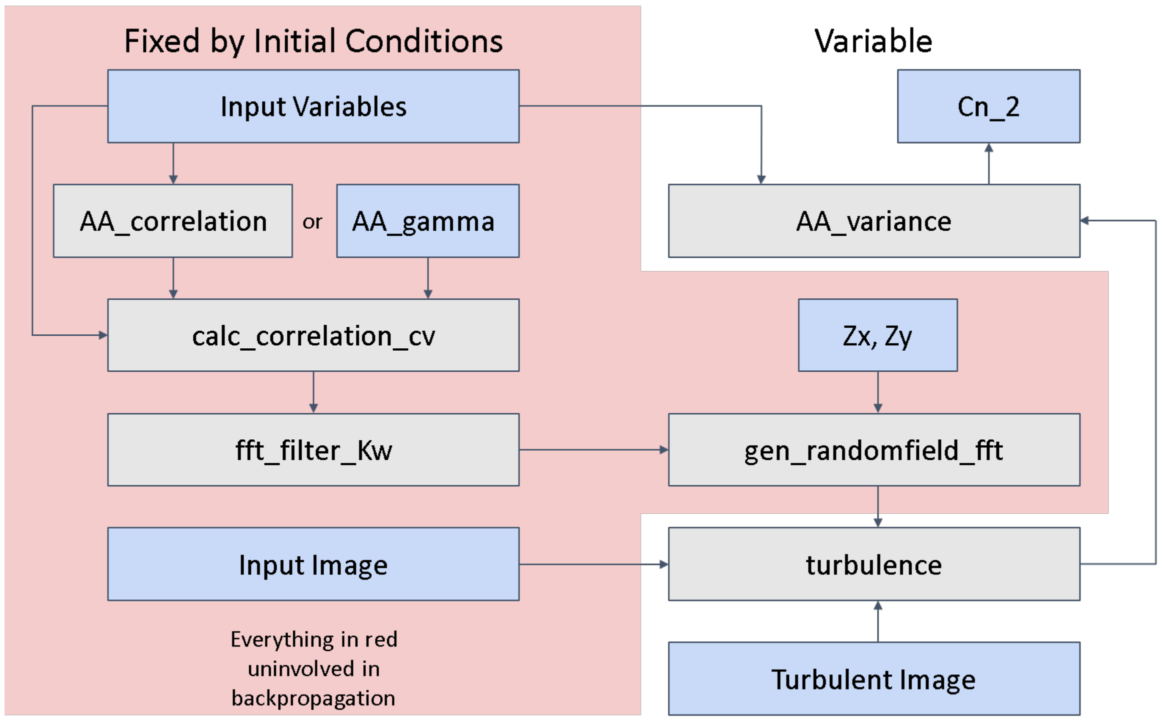


Figure 4.7: Differentiable simulator backpropagation to estimate C_n^2. The arrows that have reversed direction from Fig. 4.4 represent the direction of the error backpropagation.

easier for the machine learning stage, we initialized with the correct known white noise field(s) and added only a small degree of additional Gaussian noise. We also attempted initializing randomly the starting parameters without any prior knowledge of Z_x, Z_y, or C_n^2, as this would be a practical use case for the simulator estimating these parameters. Unfortunately, while these methods did decrease in training error over epochs, the error between the estimated white noise field(s) and true white noise field(s) consistently stagnated, often converging to an error much higher than 0, and sometimes even increasing. Additionally, the incorporation of Z_x and Z_y estimation while simultaneously estimating C_n^2 resulted in a failure to converge to the correct C_n^2 value most of the time; the estimation of C_n^2 is likely dependent heavily on having the correct white noise field.

We considered that there may be multiple white noise fields that resulted in the correct distorted image, which could account for the continuously decreasing training loss despite convergence of Z_x and Z_y to the incorrect values. Unfortunately, qualitative comparisons between the distorted images indicated that this was not the case, since the warp pattern did not appear similar for the images. Ultimately, this failure of the differentiable simulator makes theoretical sense. Due to the scale of the white noise fields (on the order of tens to hundreds of thousands of parameters), a relatively limited number and scale of mathematical operations would not be able to account for the effects of each of the individual parameters sufficiently. This is why the estimation of the warp field in the first place is performed by deep neural networks with a large capacity for function approximation. If each of the operations in the differentiable simulator were invertible, this would be a simple task, but many of the operations are not bijective, and are therefore not invertible.

However, in light of our non-rigid distortion removal network is capable of estimating a warp field, we attempted to proceed with the differentiable simulator by treating the warp field as the initial guess for the warp field within the simulator. The operations that transform the white noise vector fields Z_x, Z_y into the warp field F in the simulator are not invertible, but we aimed to estimate C_n^2 by using F in place of Z_X and Z_y and treating the estimated turbulent image from the network as the true turbulent image. This lead to a practical problem in that the warp field produced by the network already is scaled according to whatever C_n^2 value was in the images from which the network produced the warp field. We thus attempted to scale the result by C_n^2 artificially and scale back down when the error was incorporated, so that the C_n^2 would still be a parameter that the simulator's forward pass leverages, and thereby still be a parameter that can be adjusted in the backpropagation stage.

This lead to highly variable and innacurate results. Different initializations of C_n^2 sometimes resulted in the simulator converging to different local minima, and the simulator had an average percent error of approximately 100%. Naturally, this still allows for a decent ball-park estimate of C_n^2 values, but not to the degree of precision that we believed was achievable. We thus moved on from this approach in favor of using a neural network to directly estimate C_n^2 from warp fields.

4.4 Deep Learning Estimation of C_n^2

Estimating C_n^2 from warp fields using deep learning makes practical sense; the input data is large, as abundant as necessary when using simulated data to train, and in a form easily processable by a neural network. We leveraged the turbulence simulator to construct a dataset that would allow robust estimation of a variety of C_n^2 values. We generated 5000 random numbers Y to be C_n^2 values where $Y \sim 10^{\mathcal{U}(-16,-12)}$ ($\mathcal{U}(a, b)$ being the continuous uniform distribution). We then used the turbulence simulator to generated 5000 random 256x256 warp fields, each with one of the random C_n^2 values. The warp fields were treated as inputs with corresponding C_n^2 values as labels for supervised training. 4000 warp field/C_n^2 value pairs were assigned to the training dataset, and 1000 were designated as the test dataset.

We performed the same procedure drawing the 5000 random C_n^2 values where $Y \sim \mathcal{U}(10^{-16}, 10^{-12})$. This allowed us to determine which dataset would result in optimal classification, and to what degree the choice of distribution affected the outcome of the training. With both datasets, we aimed to encompass the entire range of values for C_n^2 suggested by [71].

The model uses the AlexNet architecture as defined in the Torchvision Python package, but modified to take a warp field as input. This architecture is outlined in Table 4.1. The architecture contains 5 2-dimensional convolutional layers, the first

Layer	Output Shape	Activation	# Parameters
Conv2d-1	$n, 64, 31, 31$	ReLU	15,552
MaxPool2d-1	$n, 64, 15, 15$	-	-
Conv2d-2	$n, 192, 15, 15$	ReLU	307,392
MaxPool2d-2	$n, 192, 7, 7$	-	-
Conv2d-3	$n, 384, 7, 7$	ReLU	663,936
Conv2d-4	$n, 256, 7, 7$	ReLU	884,992
Conv2d-5	$n, 256, 7, 7$	ReLU	590,080
MaxPool2d-3	$n, 256, 3, 3$	-	-
AdaptiveAvgPool2d	$n, 256, 6, 6$	-	-
Flatten	$n, 9216$	-	-
Dropout-1	$n, 9216$	-	-
FullyConnected-1	$n, 4096$	ReLU	37,752,832
Dropout-2	$n, 4096$	-	-
FullyConnected-2	$n, 4096$	ReLU	16,781,312
FullyConnected-3	$n, 1$	ReLU	4,097

Table 4.1: Architecture of the C_n^2 network with n warp fields as input. The entire network has 57,000,193 trainable parameters.

Epochs	Test Loss	% Error	r	Run Time
1,500	4.143e-5	72.044	0.9655	$\sim$ 0.75 hours
8,000	5.106e-5	18.479	0.9972	$\sim$ 4 hours
20,000	2.309e-5	14.204	0.9978	$\sim$ 10 hours

Table 4.2: Comparison of different lengths of training time for C_n^2 network. "% Error" represents the true percent error between the values, not log_{10} values. "r" represents the Pearson product-moment correlation coefficient between true outputs and estimated outputs.

two of which are followed by Max Pooling layers; all have ReLU activation functions. The first convolutional layer has a kernel size of 11, the second layer, 5, and the remaining three layers, 3. There is then an 6-by-6 adaptive average pooling layer followed by a flattening. Finally, 3 fully-connected layers, the first two preceded by dropout and all followed by ReLU activations lead to the final output vector. For training, we used the Adam optimizer with a learning rate of 1e-4 and mean squared error loss. All in all, the architecture is large, and thus training took substantial memory and time.

We trained the model in parallel on two Titan Xp GPUs. Due to the video random access memory limits of the GPUs, it was necessary to take a 128x128 region of each warp field; thus the training input was 4,000x2x128x128 and the training output was 4,000x1. When training, we discovered that of the two probability distributions from which to sample C_n^2 values to construct a training dataset, the first distribution resulted in faster convergence and higher test accuracy than the second distribution. Therefore, we data presented here was derived from the former probability distribution. Table 4.2 shows the test loss and percent error for various training lengths.

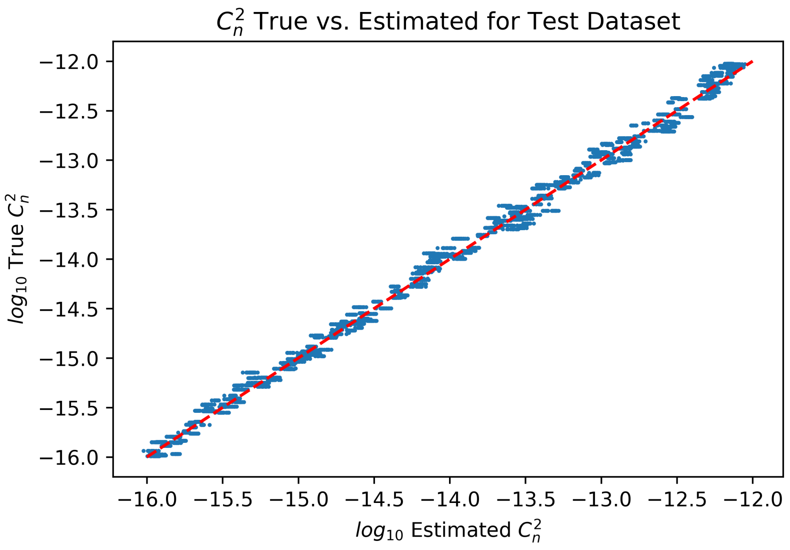


Figure 4.8: Scatter plot of true vs. estimated C_n^2 for the 1000 values in the test dataset.

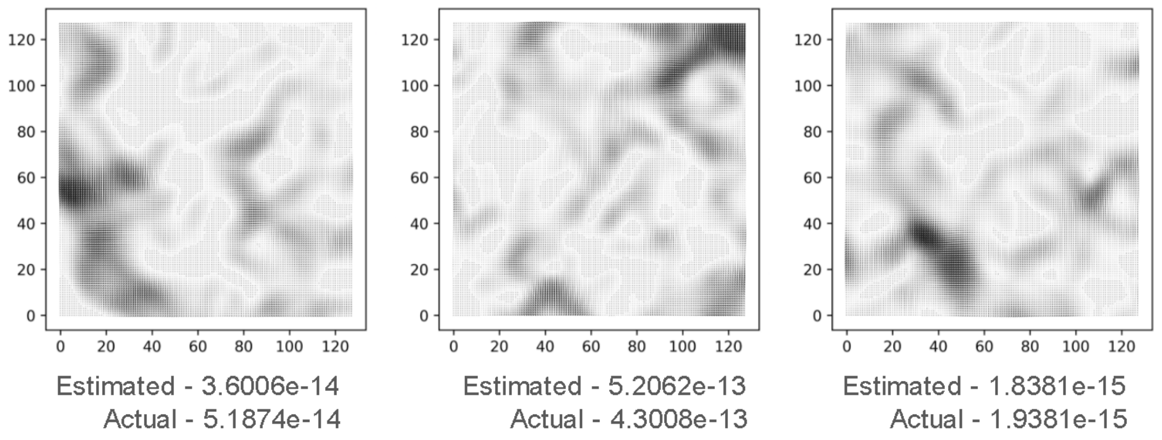


Figure 4.9: Example warp fields from test dataset across a range of C_n^2 values, with corresponding C_n^2 values shown. The warp fields were generated with the actual C_n^2 value shown, and the estimated C_n^2 was generated from the model.

The model saw substantial improvement between 1,500 epochs and 8,000 epochs, but improvements diminished, although continued, when increasing training duration to 20,000 epochs. Overall, the model demonstrated an absolute error of 14.204%, and was correct well within an order of magnitude for the entire training dataset. 4.9. Given each model's performance on its respective dataset, our model outperforms the methods presented in [77] and [63]. While [63] provides increasingly inaccurate estimations with lower C_n^2 values, and likewise [77] frequently overestimates lower C_n^2 values, Fig. 4.8 demonstrates that the true and estimated C_n^2 values are estimated accurately across the entire range of $(10e-16, 10e-12)$. Additionally, the best model in [63] reported an r value of 0.9323 between target and output, and [77] reported an r value of 0.91, whereas we achieved an r value of 0.9978. It should be noted that the test performance of our estimator is based on simulated data, whereas both of the aforementioned studies were comparing with reference instrument measurements (specifically, a sonic anemometer for [77]). We discuss approaches to demonstrate the robustness of the model in the conclusion of this thesis.

Chapter 5

CONCLUSION

In this thesis we have presented an unsupervised network that effectively removes distortion from turbulent images. The network utilizes a two-step framework that allows for estimation of both a distortion-free image and the nonrigid distortion jointly. Since the network is unsupervised, it does not suffer from the poor generalizeability as with supervised approaches; rather, it is robust to different types of turbulence. We demonstrate that the network is capable of distortion removal comparable to state-of-the-art models for both air and water turbulence, and can account for a variety of turbulent conditions in either circumstance.

Additionally, we have presented a deep-learning-based method for estimating C_n^2 directly from a warp field. We demonstrate its efficacy for estimation of a wide variety of C_n^2 values, and show that it also exceeds the performance of other deep learning methods. Unlike other methods, it requires no prior data about the turbulent conditions save those preserved within a single warp field. If we use this method in tandem with the non-rigid image distortion removal network, we could provide a set of 10 images and receive an estimation of C_n^2 from the warp field produced by the network, in addition to the estimated image already provided by that network.

5.1 Limitations and Future Directions

Our non-rigid image distortion removal network does have certain limitations. The network typically performs well on a sequence of 10 images, but this performance relies heavily on a quality initialization; it is possible that poor initializations may result from the relatively small number of images. Because the network is unsupervised, the training process requires an increased run time as a tradeoff for not requiring any initial training of the network. The network also does not incorporate known physical constraints of turbulence, and thus may be inconsistent with the physics of turbulence.

It therefore follows that incorporation of losses based on the statistics of turbulence may improve the estimation capabilities of the network. We hope to continue exploring avenues to mitigate these disadvantages and improve the network's capabilities. This may include incorporating portions of the differentiable simulator into the network. Initial attempts in this area have already been made involving making components of the physics-based 2D turbulence simulator differentiable.

Furthermore, the network does not yet effectively account for camera jitter motion or moving objects within a scene. Video stabilization techniques thus may be necessary to mitigate the scene disparities from camera jitter motion prior to training. In order to distinguish turbulent motion, object motion, and camera jitter, we may also have to extend the model to consider the differences between these types of motion, and we plan to make the model robust enough to operate well under conditions with additional types of motion.

The accuracy of our C_n^2 estimations, while good, likely could be improved. This may be accomplished by averaging the estimations from several sub-regions of a single warp field or by combining several warp fields. The latter option nicely complements the multiple warp fields produced by the distortion removal network, each of which can be assumed to have the same C_n^2 values. Additionally, we could likely find a more optimal architecture to accomplish this task. While we face an important hardware limitation in the form of finite VRAM, a newer, lean-but-effective architecture may provide a more accurate estimation than the AlexNet architecture used.

REFERENCES

[1] Anantrasirichai, N., A. Achim and D. Bull, "Atmospheric turbulence mitigation for sequences with moving objects using recursive image fusion", in "2018 25th IEEE International Conference on Image Processing (ICIP)", pp. 2895–2899 (IEEE, 2018).

[2] Aubailly, M., M. A. Vorontsov, G. W. Carhart and M. T. Valley, "Automated video enhancement from a stream of atmospherically-distorted images: the lucky-region fusion approach", in "Atmospheric Optics: Models, Measurements, and Target-in-the-Loop Propagation III", vol. 7463, p. 74630C (International Society for Optics and Photonics, 2009).

[3] Azinovic, D., T.-M. Li, A. Kaplanyan and M. Niessner, "Inverse path tracing for joint material and lighting estimation", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2447–2456 (2019).

[4] Çaliskan, T. and N. Arica, "Atmospheric turbulence mitigation using optical flow", in "2014 22nd International Conference on Pattern Recognition", pp. 883–888 (IEEE, 2014).

[5] Chen, G., Z. Gao, Q. Wang and Q. Luo, "U-net like deep autoencoders for deblurring atmospheric turbulence", Journal of Electronic Imaging **28**, 5, 053024 (2019).

[6] Chen, W., F. Wei, K. N. Kutulakos, S. Rusinkiewicz and F. Heide, "Learned feature embeddings for non-line-of-sight imaging and recognition", ACM Transactions on Graphics (TOG) (2020).

[7] Chibane, J. and G. Pons-Moll, "Implicit feature networks for texture completion from partial 3d data", in "European Conference on Computer Vision", (Springer, 2020).

[8] Chimitt, N., Z. Mao, G. Hong and S. H. Chan, "Rethinking atmospheric turbulence mitigation", arXiv preprint arXiv:1905.07498 (2019).

[9] Deledalle, C.-A. and J. Gilles, "Blind atmospheric turbulence deconvolution", IET Image Processing **14**, 14, 3422–3432 (2020).

[10] Dey, A., "Machine learning algorithms: a review", International Journal of Computer Science and Information Technologies **7**, 3, 1174–1179 (2016).

[11] Donate, A., G. Dahme and E. Ribeiro, "Classification of textures distorted by waterwaves", in "18th International Conference on Pattern Recognition (ICPR'06)", vol. 2, pp. 421–424 (IEEE, 2006).

[12] Donate, A. and E. Ribeiro, "Improved reconstruction of images distorted by water waves.", in "VISAPP (1)", pp. 228–235 (2006).

[13] Dupont, E., M. B. Martin, A. Colburn, A. Sankar, J. Susskind and Q. Shan, "Equivariant neural rendering", in "International Conference on Machine Learning", (PMLR, 2020).

[14] Efros, A., V. Isler, J. Shi and M. Visontai, "Seeing through water", in "Advances in Neural Information Processing Systems", pp. 393–400 (2005).

[15] Foias, C., O. Manley, R. Rosa and R. Temam, *Navier-Stokes equations and turbulence*, vol. 83 (Cambridge University Press, 2001).

[16] Fried, D. L., "Probability of getting a lucky short-exposure image through turbulence", JOSA **68**, 12, 1651–1658 (1978).

[17] Gao, J., N. Anantrasirichai and D. Bull, "Atmospheric turbulence removal using convolutional neural network", arXiv preprint arXiv:1912.11350 (2019).

[18] Ghahramani, Z., "Unsupervised learning", in "Summer School on Machine Learning", pp. 72–112 (Springer, 2003).

[19] Gilles, J., T. Dagobert and C. De Franchis, "Atmospheric turbulence restoration by diffeomorphic image registration and blind deconvolution", in "International Conference on Advanced Concepts for Intelligent Vision Systems", pp. 400–409 (Springer, 2008).

[20] Gkioulekas, I., S. Zhao, K. Bala, T. Zickler and A. Levin, "Inverse volume rendering with material dictionaries", ACM Transactions on Graphics (TOG) **32**, 6, 162 (2013).

[21] Gong, K., C. Catana, J. Qi and Q. Li, "Pet image reconstruction using deep image prior", IEEE transactions on Medical Imaging **38**, 7, 1655–1665 (2018).

[22] Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning* (MIT Press, 2016), `http://www.deeplearningbook.org`.

[23] Graupe, D., *Principles of artificial neural networks*, vol. 7 (World Scientific, 2013).

[24] Gu, J., Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks", Pattern Recognition **77**, 354–377 (2018).

[25] Halder, K. K., M. Tahtali and S. G. Anavatti, "Moving object detection and tracking in videos through turbulent medium", Journal of Modern Optics **63**, 11, 1015–1021 (2016).

[26] He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 770–778 (2016).

[27] He, R., Z. Wang, Y. Fan and D. Fengg, “Atmospheric turbulence mitigation based on turbulence extraction”, in “2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)”, pp. 1442–1446 (IEEE, 2016).

[28] Hirsch, M., S. Sra, B. Schölkopf and S. Harmeling, “Efficient filter flow for space-variant multiframe blind deconvolution”, in “2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition”, pp. 607–614 (IEEE, 2010).

[29] Hubel, D. H. and T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex”, The Journal of physiology **195**, 1, 215–243 (1968).

[30] Huebner, C. S., “Turbulence mitigation of short exposure image data using motion detection and background segmentation”, in “Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXIII”, vol. 8355, p. 83550I (International Society for Optics and Photonics, 2012).

[31] John, S. and M. A. Vorontsov, “Multiframe selective information fusion from robust error estimation theory”, IEEE Transactions on Image Processing **14**, 5, 577–584 (2005).

[32] Jordan, M. I. and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects”, Science **349**, 6245, 255–260 (2015).

[33] Kolmogorov, A. N., “Dissipation of energy in the locally isotropic turbulence”, in “Dokl. Akad. Nauk SSSR A”, vol. 32, pp. 16–18 (1941).

[34] Kolmogorov, A. N., “The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers”, Cr Acad. Sci. URSS **30**, 301–305 (1941).

[35] LeCun, Y., Y. BengiNo and G. Hinton, “Deep learning”, Nature **521**, 436–444, URL `https://doi.org/10.1038/nature14539` (2015).

[36] LeCun, Y., B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard and L. D. Jackel, “Handwritten digit recognition with a back-propagation network”, in “Advances in neural information processing systems”, pp. 396–404 (1990).

[37] LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE **86**, 11, 2278–2324 (1998).

[38] Levin, I. M., V. V. Savchenko and V. J. Osadchy, “Correction of an image distorted by a wavy water surface: laboratory experiment”, Applied Optics **47**, 35, 6650–6655 (2008).

[39] Li, T.-M., M. Aittala, F. Durand and J. Lehtinen, “Differentiable monte carlo ray tracing through edge sampling”, ACM Trans. Graph. (Proc. SIGGRAPH Asia) **37**, 6, 222:1–222:11 (2018).

[40] Li, Z., Z. Murez, D. Kriegman, R. Ramamoorthi and M. Chandraker, “Learning to see through turbulent water”, in “Winter Conference on Applications of Computer Vision”, pp. 512–520 (2018).

[41] Liu, J., Y. Sun, X. Xu and U. S. Kamilov, “Image restoration using total variation regularized deep image prior”, in “ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)”, pp. 7715–7719 (IEEE, 2019).

[42] Liu, S., T. Li, W. Chen and H. Li, “Soft rasterizer: A differentiable renderer for image-based 3d reasoning”, in “Proceedings of the IEEE International Conference on Computer Vision”, pp. 7708–7717 (2019).

[43] Mao, Z., N. Chimitt and S. H. Chan, “Image reconstruction of static and dynamic scenes through anisoplanatic turbulence”, IEEE Transactions on Computational Imaging **6**, 1415–1428 (2020).

[44] Mildenhall, B., P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis”, in “ECCV”, (2020).

[45] Nieuwenhuizen, R., J. Dijk and K. Schutte, “Dynamic turbulence mitigation for long-range imaging in the presence of large moving objects”, EURASIP Journal on Image and Video Processing **2019**, 1, 2 (2019).

[46] Nieuwenhuizen, R. and K. Schutte, “Deep learning for software-based turbulence mitigation in long-range imaging”, in “Artificial Intelligence and Machine Learning in Defense Applications”, vol. 11169, p. 111690J (International Society for Optics and Photonics, 2019).

[47] Nieuwenhuizen, R. P., A. W. van Eekeren, J. Dijk and K. Schutte, “Dynamic turbulence mitigation with large moving objects”, in “Electro-Optical and Infrared Systems: Technology and Applications XIV”, vol. 10433, p. 104330S (International Society for Optics and Photonics, 2017).

[48] Nimier-David, M., D. Vicini, T. Zeltner and W. Jakob, “Mitsuba 2: A retargetable forward and inverse renderer”, Transactions on Graphics (Proceedings of SIGGRAPH Asia) **38**, 6 (2019).

[49] Oreifej, O., X. Li and M. Shah, “Simultaneous video stabilization and moving object detection in turbulence”, IEEE Transactions on Pattern Analysis and Machine Intelligence **35**, 2, 450–462 (2012).

[50] Oreifej, O., G. Shu, T. Pace and M. Shah, “A two-stage reconstruction approach for seeing through water”, in “CVPR 2011”, pp. 1153–1160 (IEEE, 2011).

[51] Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, “Automatic differentiation in pytorch”, (2017).

[52] Rahaman, N., A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio and A. Courville, "On the spectral bias of neural networks", in "International Conference on Machine Learning", pp. 5301–5310 (PMLR, 2019).

[53] Rasouli, S. and M. T. Tavassoly, "Measurement of the refractive-index structure constant, c2n, and its profile in the ground level atmosphere by moiré technique", in "Optics in Atmospheric Propagation and Adaptive Systems IX", vol. 6364, p. 63640G (International Society for Optics and Photonics, 2006).

[54] Ren, D., K. Zhang, Q. Wang, Q. Hu and W. Zuo, "Neural blind deconvolution using deep priors", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 3341–3350 (2020).

[55] Riegler, G. and V. Koltun, "Free view synthesis", in "European Conference on Computer Vision", pp. 623–640 (Springer, 2020).

[56] Roadcap, J. R. and P. Tracy, "A preliminary comparison of daylit and night c 2 n profiles measured by thermosonde", Radio Science **44**, 02, 1–8 (2009).

[57] Roggemann, M. C., C. A. Stoudt and B. M. Welsh, "Image-spectrum signal-to-noise-ratio improvements by statistical frame selection for adaptive-optics imaging through atmospheric turbulence", Optical Engineering **33**, 10, 3254–3265 (1994).

[58] Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge", International journal of computer vision **115**, 3, 211–252 (2015).

[59] Schaffer, J. D., D. Whitley and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art", in "[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks", pp. 1–37 (IEEE, 1992).

[60] Schwartzman, A., M. Alterman, R. Zamir and Y. Y. Schechner, "Turbulence-induced 2d correlated image distortion", in "2017 IEEE International Conference on Computational Photography (ICCP)", pp. 1–13 (IEEE, 2017).

[61] Shimizu, M., S. Yoshimura, M. Tanaka and M. Okutomi, "Super-resolution from image sequence under influence of hot-air optical turbulence", in "2008 IEEE Conference on Computer Vision and Pattern Recognition", pp. 1–8 (IEEE, 2008).

[62] Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556 (2014).

[63] Su, C., X. Wu, T. Luo, S. Wu and C. Qing, "Adaptive niche-genetic algorithm based on backpropagation neural network for atmospheric turbulence forecasting", Applied optics **59**, 12, 3699–3705 (2020).

[64] Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, “Going deeper with convolutions”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 1–9 (2015).

[65] Tancik, M., P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron and R. Ng, “Fourier features let networks learn high frequency functions in low dimensional domains”, NeurIPS (2020).

[66] Thapa, S., N. Li and J. Ye, “Dynamic fluid surface reconstruction unsing deep neural network”, in “Conference on Computer Vision and Pattern Recognition”, (2020).

[67] Tian, Y. and S. G. Narasimhan, “Seeing through water: Image restoration using model-based tracking”, in “2009 IEEE 12th International Conference on Computer Vision”, pp. 2303–2310 (IEEE, 2009).

[68] Tian, Y. and S. G. Narasimhan, “A globally optimal data-driven approach for image distortion estimation”, in “2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition”, pp. 1277–1284 (IEEE, 2010).

[69] Tian, Y., S. G. Narasimhan and A. J. Vannevel, “Depth from optical turbulence”, in “2012 IEEE Conference on Computer Vision and Pattern Recognition”, pp. 246–253 (IEEE, 2012).

[70] Tsai, C.-Y., A. C. Sankaranarayanan and I. Gkioulekas, “Beyond volumetric albedo — A surface optimization framework for non-line-of-sight imaging”, in “IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR)”, (2019).

[71] Tunick, A., N. Tikhonov, M. Vorontsov and G. Carhart, “Characterization of optical turbulence (cn2) data measured at the arl a_lot facility”, Tech. rep., US Army Research Laboratory Adelphi United States (2005).

[72] Ulyanov, D., A. Vedaldi and V. Lempitsky, “Deep image prior”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 9446–9454 (2018).

[73] Vint, D., G. Di Caterina, J. Soraghan, R. Lamb and D. Humphreys, “Analysis of deep learning architectures for turbulence mitigation in long-range imagery”, in “Artificial Intelligence and Machine Learning in Defense Applications II”, vol. 11543, p. 1154303 (International Society for Optics and Photonics, 2020).

[74] Vorontsov, A. M., M. A. Vorontsov, G. A. Filimonov and E. Polnau, “Atmospheric turbulence study with deep machine learning of intensity scintillation patterns”, Applied Sciences **10**, 22, 8136 (2020).

[75] Vorontsov, M. A. and G. W. Carhart, “Anisoplanatic imaging through turbulent media: image recovery by local information fusion from a set of short-exposure images”, JOSA A **18**, 6, 1312–1324 (2001).

[76] Voynov, O., A. Artemov, V. Egiazarian, A. Notchenko, G. Bobrovskikh, E. Burnaev and D. Zorin, “Perceptual deep depth super-resolution”, in “Proceedings of the IEEE International Conference on Computer Vision”, pp. 5653–5663 (2019).

[77] Wang, Y. and S. Basu, “Using an artificial neural network approach to estimate surface-layer optical turbulence at mauna loa, hawaii”, Optics letters **41**, 10, 2334–2337 (2016).

[78] Wen, Z., D. Fraser, A. Lambert and H. Li, “Reconstruction of underwater image by bispectrum”, in “2007 IEEE International Conference on Image Processing”, vol. 3, pp. III–545 (IEEE, 2007).

[79] Werbos, P. J., “Backpropagation through time: what it does and how to do it”, Proceedings of the IEEE **78**, 10, 1550–1560 (1990).

[80] Xu, L., S. Zheng and J. Jia, “Unnatural l0 sparse representation for natural image deblurring”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 1107–1114 (2013).

[81] Xue, T., M. Rubinstein, N. Wadhwa, A. Levin, F. Durand and W. T. Freeman, “Refraction wiggles for measuring fluid depth and velocity from video”, in “European Conference on Computer Vision”, pp. 767–782 (Springer, 2014).

[82] Zhang, C., L. Wu, C. Zheng, I. Gkioulekas, R. Ramamoorthi and S. Zhao, “A differential theory of radiative transfer”, ACM Transactions on Graphics (TOG) **38**, 6, 1–16 (2019).

[83] Zhang, C., B. Xue, F. Zhou and W. Xiong, “Removing atmospheric turbulence effects in unified complex steerable pyramid framework”, IEEE Access **6**, 75855–75867 (2018).

[84] Zhang, M., X. Lin, M. Gupta, J. Suo and Q. Dai, “Recovering scene geometry under wavy fluid via distortion and defocus analysis”, in “European Conference on Computer Vision”, pp. 234–250 (Springer, 2014).

[85] Zhang, X., S. Fanello, Y.-T. Tsai, T. Sun, T. Xue, R. Pandey, S. Orts-Escolano, P. Davidson, C. Rhemann, P. Debevec *et al.*, “Neural light transport for relighting and view synthesis”, ACM Transactions on Graphics (TOG) (2021).

[86] Zhu, X., “Measuring spatially varying blur and its application in digital image restoration”, (2013).

[87] Zhu, X. and P. Milanfar, “Image reconstruction from videos distorted by atmospheric turbulence”, in “Visual Information Processing and Communication”, vol. 7543, p. 75430S (International Society for Optics and Photonics, 2010).

[88] Zhu, X. and P. Milanfar, “Removing atmospheric turbulence via space-invariant deconvolution”, IEEE transactions on pattern analysis and machine intelligence **35**, 1, 157–170 (2012).