8-18-2021

Learning to Interpret Fluid Type Phenomena via Images

Simron Thapa
Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_dissertations

Part of the Computer Sciences Commons

Recommended Citation
https://digitalcommons.lsu.edu/gradschool_dissertations/5638

This Dissertation is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Doctoral Dissertations by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.
Learning to Interpret Fluid Type Phenomena via Images

Simron Thapa

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_dissertations

Part of the Computer Sciences Commons
LEARNING TO INTERPRET FLUID TYPE PHENOMENA VIA IMAGES

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Division of Computer Science and Engineering

by

Simron Thapa
M.S. Computer Science, Louisiana State University, 2017
December 2021
To my grandparents,

and my parents,

Ganesh Thapa and Sharmila K. Thapa,

for making all of this possible.
Acknowledgments

First, and most of all, I would like to express my heartfelt gratitude to my advisor Dr. Jinwei Ye for her guidance and encouragement in every aspect of my education at LSU. Without her support and invaluable suggestions, this thesis and my academic, as well as intellectual growth, would have not been possible. I am also very thankful to Dr. Bijaya B. Karki for being on my committee and for his constant support throughout my time at LSU. I am thankful for his continued encouragement and guidance as I embark on my academic journey ahead. I am also extremely grateful to have Dr. Jianhua Chen as my Ph.D. thesis committee member. Thank you for your expertise, vision, and heartening motivation. I would also like to thank Dr. Gerald M. Knapp for his thorough examination of my work and helpful comments. I will forever cherish these invaluable pieces of advice. A special thanks to Dr. Suren Jaysuriya for the rewarding experience I got from the collaboration with him and the Imaging Lyceum Lab at Arizona State University.

I’d like to thank Dr. Nianyi Li for her guidance, motivation, and thoughtful inputs throughout my research journey. It just so happens that I am taking the credit as the sole author of this dissertation, but none of this work would have been possible without your collaborative efforts and continual encouragement. I’d also like to appreciate my fellow lab mate, Yuqi Ding for all the stimulating discussions in the past four years. Special thank you to my close friend Danissa Rodriguez and all my friends for making my stay at LSU a memory of a lifetime. Each of you made this journey easier for me in many ways.

Finally, this journey would not have been possible without the continual and unconditional love and support from my family. So, thank you mom (Sharmila), dad (Ganesh) and brother (Samyam), for believing in me and their continuous motivation to pursue my dreams and succeed in my career path. A special thanks to Bijay, Santosh, Sarala, Anjila, Jaya, Anil, and Sanjita for being positive figures in my life and providing me with words of encouragement during hard times. Thank you Sagar for sticking with me through all the thick and thins of my journey. I am blessed to be surrounded by these positive souls.
Table of Contents

ACKNOWLEDGMENTS ................................................................. iii
LIST OF TABLES ................................................................. vi
LIST OF FIGURES ............................................................... vii
ABSTRACT ........................................................................... ix

CHAPTER

1 INTRODUCTION ................................................................. 1
  1.1 Dynamic Fluid Surface Reconstruction ................................. 2
  1.2 Underwater Image Restoration ........................................... 3
  1.3 Non-rigid Image Distortion Removal ..................................... 4
  1.4 Challenges .................................................................... 5
  1.5 Contributions ................................................................ 7
  1.6 Blueprint of the Dissertation ............................................. 9

2 DEEP NEURAL NETWORK FOR DYNAMIC FLUID SURFACE
   RECONSTRUCTION .............................................................. 11
  2.1 Introduction .................................................................... 11
  2.2 Related Work .................................................................. 13
  2.3 Network Design ............................................................. 14
  2.4 Experiments .................................................................... 24
  2.5 Conclusions ................................................................... 33

3 LEARNING TO REMOVE REFRACTIVE DISTORTIONS
   FROM UNDERWATER IMAGE .............................................. 36
  3.1 Introduction .................................................................... 36
  3.2 Related Work .................................................................. 38
  3.3 Network Design ............................................................. 40
  3.4 Experiments .................................................................... 47
  3.5 Conclusion ...................................................................... 65

4 UNSUPERVISED NON-RIGID IMAGE DISTORTION REMOVAL
   VIA GRID DEFORMATION .................................................. 66
  4.1 Introduction .................................................................... 66
  4.2 Related Work .................................................................. 68
  4.3 Network Design ............................................................. 70
  4.4 Experiments .................................................................... 76
  4.5 Conclusions .................................................................... 88

5 A COMBINED APPROACH TO SOLVING FLUID TYPE
   PHENOMENA VIA IMAGES ............................................. 90
  5.1 Introduction .................................................................... 90
**List of Tables**

2.1 Detailed network architecture of FSRN-CNN. ........................................ 17
2.2 Detailed network architecture of FSRN-RNN. ........................................ 18
2.3 Quantitative comparison with existing methods on depth estimation. .......... 26
2.4 Depth and surface normal estimation measurements for ablation study. ...... 28
3.1 Detailed architecture of the Dis-Net. .................................................... 49
3.2 Detailed architecture of the DG-GAN. .................................................. 50
3.3 Quantitative comparison with the state-of-the-arts. ............................... 58
3.4 Quantitative ablation on physics-based loss terms. ................................. 62
3.5 Quantitative ablation on the temporal consistency. ................................. 64
4.1 Architecture of grid deformer $G_{\theta}$ and the image generator $I_{\phi}$. .. 72
4.2 Air turbulence simulation parameters. .................................................... 78
4.3 Quantitative comparison on air turbulence data with various strengths....... 82
4.4 Quantitative comparison on different types of water turbulence. .............. 84
4.5 Comparison of the performance on the restoration ability of $G_{\theta}$. .......... 87
4.6 Average PSNR, SSIM, and running time (2,000 epochs). .......................... 87
4.7 Quantitative comparison on varied Fourier feature mapping. .................... 89
# List of Figures

2.1 Our dynamic fluid surface reconstruction scheme. ........................................... 11  
2.2 The setting of our fluid surface reconstruction problem. ................................. 14  
2.3 The workflow of our FSRN. ........................................................................... 16  
2.5 Sample images from our fluid dataset. .............................................................. 22  
2.6 Fluid surface reconstruction on synthetic data. .................................................. 26  
2.7 Visual comparison with existing methods on depth estimation. ........................ 27  
2.8 Visual comparisons for the ablation study. ....................................................... 29  
2.9 Our experimental setup for real data acquisition. .............................................. 30  
2.10 Reconstruction results on real data................................................................. 31  
2.11 Comparison between FSRN-CNN and our full network FSRN. ......................... 31  
2.12 Re-render refraction image using our reconstructed fluid surface. ................. 32  
2.14 Additional results on real data. ....................................................................... 33  
2.4 The overall architecture of our FSRN. ............................................................. 35  
2.13 Additional results on synthetic data. ............................................................... 36  
2.1 Our distortion guided network for underwater image correction. .................... 37  
3.2 Our network results on real drone scene. ......................................................... 38  
3.3 The overall architecture of DG-Net. ................................................................. 39  
3.4 Illustration of refractive distortion. ................................................................... 43  
3.5 The network architecture of a CNN branch of our Dis-Net. .............................. 44  
3.6 Illustration of forward $\mathcal{F}$ and backward mapping $\mathcal{B}$. ....................... 46  
3.7 Sample images of our synthetic underwater image dataset. ............................. 47  
3.8 More sample images of our synthetic underwater image dataset. ..................... 54  
3.9 Visual comparison with [71] on the SynSet. .................................................... 55  
3.10 Visual comparison on the real captured TianSet [112] ..................................... 55
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.11</td>
<td>Visual comparative results on real ThapaSet[111]</td>
<td>56</td>
</tr>
<tr>
<td>3.12</td>
<td>Qualitative ablation on physics-based loss terms.</td>
<td>57</td>
</tr>
<tr>
<td>3.13</td>
<td>More visual comparative results on real underwater image dataset.</td>
<td>59</td>
</tr>
<tr>
<td>3.14</td>
<td>Comparison on the “in the wild” dataset.</td>
<td>60</td>
</tr>
<tr>
<td>3.15</td>
<td>Visual results on Pool Scene taken with drone.</td>
<td>61</td>
</tr>
<tr>
<td>3.16</td>
<td>Visual results on aquarium scene.</td>
<td>63</td>
</tr>
<tr>
<td>3.17</td>
<td>Refraction images with different distortion levels.</td>
<td>65</td>
</tr>
<tr>
<td>3.18</td>
<td>Comparisons with respect to the distortion levels.</td>
<td>65</td>
</tr>
<tr>
<td>4.1</td>
<td>Our unsupervised network for distortion-free prediction.</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>Our unsupervised non-rigid image distortion removal network.</td>
<td>68</td>
</tr>
<tr>
<td>4.3</td>
<td>Distorted image generation via grid deformation.</td>
<td>73</td>
</tr>
<tr>
<td>4.4</td>
<td>Analysis of initialization step.</td>
<td>76</td>
</tr>
<tr>
<td>4.5</td>
<td>Air turbulence simulation setup.</td>
<td>78</td>
</tr>
<tr>
<td>4.6</td>
<td>Exemplary images of the three turbulent strength levels.</td>
<td>79</td>
</tr>
<tr>
<td>4.7</td>
<td>Exemplary turbulence images of the three types of water waves.</td>
<td>81</td>
</tr>
<tr>
<td>4.8</td>
<td>Comparisons on the Building and the Chimney.</td>
<td>82</td>
</tr>
<tr>
<td>4.9</td>
<td>Visual comparison on our real captured atmospheric turbulence images.</td>
<td>82</td>
</tr>
<tr>
<td>4.10</td>
<td>Visual comparisons on real water turbulence images.</td>
<td>83</td>
</tr>
<tr>
<td>4.11</td>
<td>Qualitative comparison on simulated air turbulence images.</td>
<td>85</td>
</tr>
<tr>
<td>4.12</td>
<td>Qualitative comparison on simulated water turbulence images.</td>
<td>86</td>
</tr>
<tr>
<td>4.13</td>
<td>Ablation study on different variants of our proposed network.</td>
<td>88</td>
</tr>
<tr>
<td>4.14</td>
<td>Analysis of $\kappa_T$ and different numbers of input images.</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>The overall architecture.</td>
<td>91</td>
</tr>
<tr>
<td>5.2</td>
<td>Our solution to dynamic and transparent fluid type phenomena.</td>
<td>92</td>
</tr>
<tr>
<td>5.3</td>
<td>Real experimental results.</td>
<td>94</td>
</tr>
</tbody>
</table>
Abstract

Learning to interpret fluid-type phenomena via images is a long-standing challenging problem in computer vision. The problem becomes even more challenging when the fluid medium is highly dynamic and refractive due to its transparent nature. Here, we consider imaging through such refractive fluid media like water and air. For water, we design novel supervised learning-based algorithms to recover its 3D surface as well as the highly distorted underground patterns. For air, we design a state-of-the-art unsupervised learning algorithm to predict the distortion-free image given a short sequence of turbulent images. Specifically, we design a deep neural network that estimates the depth and normal maps of a fluid surface by analyzing the refractive distortion of a reference background pattern. Regarding the recovery of severely downgraded underwater images due to the refractive distortions caused by water surface fluctuations, we present the distortion-guided network (DG-Net) for restoring distortion-free underwater images. The key idea is to use a distortion map to guide network training. The distortion map models the pixel displacement caused by water refraction. Furthermore, we present a novel unsupervised network to recover the latent distortion-free image. The key idea is to model non-rigid distortions as deformable grids. Our network consists of a grid deformer that estimates the distortion field and an image generator that outputs the distortion-free image. By leveraging the positional encoding operator, we can simplify the network structure while maintaining fine spatial details in the recovered images. We also develop a combinational deep neural network that can simultaneously perform recovery of the latent distortion-free image as well as 3D reconstruction of the transparent and dynamic fluid surface. Through extensive experiments on simulated and real captured fluid images, we demonstrate that our proposed deep neural networks outperform the current state-of-the-art on solving specific tasks.
Chapter 1.
Introduction

There is not in the wide world a valley so sweet
As that vale in whose bosom the bright waters meet;
Oh! the last rays of feeling and life must depart,
Ere the bloom of that valley shall fade from my heart.

-- Thomas Moore
_Irish Melodies_

Water has fascinated mankind since the earliest times. It is more than just a necessity of
life; water has inspired art, poetry, myth and science. Thales the ancient Greek philosopher
described water as the primary principle, or the foundation of all matter where he explained
how all things could come into being from water, and return ultimately to the originating
material. The polymorphism of water and its optical magnificence demands amazement
and often apprehension upon the high seas. It inspired Poseidon, an ancient Greek god,
lending him the ability to change shape at will, thus ruling the sea.

This thesis engages the problem of capturing the shape of dynamic water from images
and to extract the surface underneath the dynamic fluid. Learning to interpret fluid type
phenomena via images is a long-standing challenging problem in computer vision. The
problem becomes even more challenging when the fluid medium is highly dynamic, turbu-
lent and refractive due to transparency. Reconstructing the 3D surface of such dynamic
medium and recovering the scene information from the highly refracted and turbulent im-
ages are well known problems in the area of computer vision. By far, nearly all previous
approaches have focused on using dedicated laboratory imaging system to solve these prob-
lems. We however, resort to learning based solutions to these problems and develop three
novel networks.

In this dissertation, we first explore how to recover the 3D fluid surface using a small
sequence of monocular input images and present the very first convolutional deep neural
network to solve the task at hand. We then explore recovery of underwater images which are highly distorted due to refractions in the fluctuating water. We further design our second deep neural network to perform the recovery. Finally, we extend the problem to both air and water medium and design an unsupervised neural network to recover images with general non-rigid distortions and tested on real captured images across turbulent air or water.

1.1 Dynamic Fluid Surface Reconstruction

Dynamic fluid phenomena are common in our environment. Accurate 3D reconstruction of the fluid surface helps advance our understanding of the presence and dynamics of the fluid phenomena and thus benefits many scientific and engineering fields ranging from hydraulics and hydrodynamics [8, 56] to 3D animation and visualization [34]. However, it is difficult to tackle this problem with non-intrusive image-based methods as the captured images are often severely distorted by the refraction of light that happens at the fluid-air interface. This is because to exact invariant and reliable image features under distortion is highly challenging. Further, the dynamic nature of fluid flow makes this problem even more challenging as we need to recover a sequence of 3D surfaces that are consistent both spatially and temporally to represent the fluid motion.

Classical image-based methods for recovering the 3D fluid surface typically place a known pattern at the bottom of the fluid body and use a single or multiple cameras to capture the reference pattern through the fluid flow. Pattern distortions over time or among multiple viewpoints are analyzed for 3D fluid surface reconstruction. Since a single viewpoint is under-constrained, single image-based methods often impose additional surface assumptions (e.g., planarity [4, 13, 35], integrability[125, 130], and known average height [80, 92] etc.). Otherwise, dedicated imaging systems or special optics (e.g., Bokode [130] and light field probe [59, 125]) need to be used. Multi-view approaches rely on the photo-consistency among different viewpoints to perform 3D reconstruction. The seminal work of Morris and Kutulakos [80] extends the traditional two-view geometry to refractive medium.
with single deflection assumption. Camera arrays [26] are further adopted for more robust and accurate reconstruction. As being heavily dependent on the acquisition system, these classical methods usually use costly equipment that requires much effort to build and calibrate. Applications of these methods are thus limited.

Here, we present a deep neural network for dynamic fluid reconstruction from refraction images. We use a convolutional network for depth and normal estimation, and a recurrent network for enforcing the temporal consistency of the dynamic fluid. We consider the depth-normal consistency and the physics of refraction in our loss functions for training. We also create a large fluid dataset using physics-based fluid modeling and rendering. Through both synthetic and real experiments, we have shown that our network can recover fluid surfaces with high accuracy.

1.2 Underwater Image Restoration

Underwater scenes, when observed in air, suffer from strong distortion artifacts due to the refraction through the flowing wavy water surface. Restoring the true underwater images by removing the refractive distortions can benefit numerous tasks in underwater exploration and outer-space expedition (by extending to remove the atmospheric distortions). However, it is non-trivial to remove the refractive distortions because 1) the geometric deformations are highly non-rigid and discontinuous due to the non-linear light transport through the wavy water surface; and 2) fast-evolving waves also cause blurriness in the image. Classical approaches usually take a long sequence of images (or video) of a static underwater scene, and rely on the mean/median images [101, 88] or the “lucky patches” (the patch that happens to be free from distortion in a certain frame) [36, 31] to restore the latent distortion-free image. As these methods require video input of a static scene, they cannot be used for images captured on a moving platform (for example, an underwater vehicle). The seminal work of Tian and Narasimhan[112] presents a model-based tracking method to undistort underwater images. But their parametric model cannot be easily tuned and applied to arbitrary waves. Most recently, Li et al. [71] propose a learning-based
method to correct refractive distortions using a single image. This work demonstrates great potential of using deep neural networks to tackle the challenging problem of refractive distortion removal. But the proposed network doesn’t account for physics-based constraints and requires a large training set (over 300k images from the ImageNet [25]).

Here, we present the distortion-guided network (DG-Net) for restoring distortion-free underwater images. The key idea is to use a distortion map to guide network training. The distortion map models the pixel displacement caused by water refraction. We first use a physically constrained convolutional network to estimate the distortion map from the refracted image. We then use a generative adversarial network guided by the distortion map to restore the sharp distortion-free image. Since the distortion map indicates correspondences between the distorted image and the distortion-free one, it guides the network to make better predictions. We evaluate our network on several real and synthetic underwater image datasets and show that it out-performs the state-of-the-art algorithms, especially in presence of large distortions.

1.3 Non-rigid Image Distortion Removal

Further, imaging through turbulent refractive medium (e.g., hot air, in-homogeneous gas, fluid flow) is challenging, since the non-linear light transport through the medium (e.g., refraction and scattering) causes non-rigid distortions in perceived images. However, most computer vision algorithms rely on sharp and distortion-free images to achieve the expected performance. Removal of these non-rigid image distortions is therefore critical and beneficial for many vision applications, from segmentation to recognition. 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). Water turbulence distortion, in contrast, is induced by the refraction of light at the water-air interface. Although these two types of distortions share certain visual similarities, they are fundamentally different as they are induced by different physical mechanisms. 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 [64, 65]) 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 [112, 137]. Recently, several learning-based methods are separately proposed to enhance either the air [37, 84] or the water [71] turbulent images. These methods typically require training on a large labeled dataset. Since it is difficult to obtain real turbulent images with ground truth sharp references, these methods use simulated images to augment their datasets and bootstrap the learning.

Here, we present a novel unsupervised network to recover the latent distortion-free image. The key idea is to model non-rigid distortions as deformable grids. Our network consists of a grid deformer that estimates the distortion field and an image generator that outputs the distortion-free image. By leveraging the positional encoding operator, we can simplify the network structure while maintaining fine spatial details in the recovered images. Our method doesn’t need to be trained on labeled data and has good transferability across various turbulent image datasets with different types of distortions. Extensive experiments on both simulated and real-captured turbulent images demonstrate that our method can remove both air and water distortions without much customization.

1.4 Challenges

1.4.1 3D reconstruction from single view

Reconstructing a sensible 3D structure from a single 2D image is inherently ambiguous, and technically ill-posed problem. Given an image, an infinite number of possible world scenes may have produced it. Of course most of these are physically implausible for real-world spaces, and thus the depth may still be predicted with considerable accuracy [33]. Furthermore, the problem becomes even more challenging when we are dealing with dynamic and transparent fluid scenes. It is difficult to tackle this problem with non-intrusive image-based methods for the following reasons.
• Fluids are transparent and do not have their own colors. They acquire their colors from surrounding backgrounds. It is difficult to acquire the surface structure of such view dependent surfaces [92].

• Tracing the light path involved in fluid surface reconstruction is non-trivial because of the non-linearity inherent in refraction. The non-linearity becomes even more severe in highly fluctuating fluid cases.

• Compared to static transparent objects, accurately reconstructing dynamic wavy fluid surfaces is even harder. Especially, when a sequence of few monocular images (≈3) are taken as input.

• A large scale dataset is required for the learning based 3D surface reconstruction tasks. Even though there are popular dataset available such as NYU V2 [22] and KITTI [38] dataset. They are limited to reconstruction of lambertian objects and scenes. There is lack of such large dataset for training the fluid networks and solving the transparent fluid problems.

1.4.2 Underwater image recovery

The problem of recovering faithful underwater images is critical in underwater imaging scenarios. In the following we see the specific challenges in tackling this problem.

• The geometric deformations are highly nonrigid and discontinuous due to the non-linear light transport through the wavy water surface.

• Fast-evolving waves cause blurriness in the image.

• Early solutions of taking the mean/median or “lucky patches” of a long distorted image sequence to approximate the latent distortion-free image do not consider dynamic scene (for example, an underground moving vehicle).

• Existing parametric models [112] cannot be easily tuned and applied to arbitrary waves.
• Existing deep neural network for tackling this problem doesn’t account for physics-based constraints and requires a large training set (over 300k images from the ImageNet [25]).

1.4.3 Air turbulence image recovery

The problem of recovering faithful areal turbulence image is even more challenging due to the randomness of the areal motion. It is necessary to understand the physics of air turbulence to accurately recover such images. In the following we see the specific challenges in tackling this problem.

• Light rays in air media are scattered due to media such as aerosols, smoke particulates, fog which causes blur and haze effects. This makes the distortion-free image recovery very challenging.

• Existing methods that leverage optical flow, lucky region fusion, and blind deconvolution have artifacts when reconstructing dynamic scenes with large amounts of motion.

• Existing deep neural networks for air turbulence removal are trained with synthetic or semi-synthetic data and have trouble with generalization outside of the training data (as do most supervised neural networks).

1.5 Contributions

This dissertation makes the following contributions in computer vision.

1.5.1 Dynamic Fluid Surface Reconstruction

• We presents a learning-based single image approach for 3D fluid surface reconstruction. We specifically design a deep neural network that estimates the depth and normal maps of a fluid surface by analyzing the refractive distortion of a reference background pattern. The recurrent layers of our network maintains the temporal consistency of the predicted fluid results.
• Due to the lack of fluid data, we synthesize a large fluid dataset using physics-based fluid modeling and rendering techniques for network training and validation. Our synthetic dataset consist of ground truth fluid images with their corresponding water surface depth and normal map. Here, our dataset targets to model wide variety of fluid motion like: Shallow water waves, Ocean waves, several Gaussian and Sinusoidal waves. Our dataset contains over 45,000 refraction images (75 fluid sequences) with the groundtruth depth and normal maps. We also use a variety of reference patterns (<15) to enrich our dataset, which include noise patterns (Perlin, Simplex, and Worley), checkerboards with different sizes, and miscellaneous textures (bricks, tiles, etc). We also generate real images sequences in laboratory setting for testing purposes.

• Through experiments on simulated and real captured fluid images, we demonstrate that our proposed deep neural network trained on our fluid dataset can recover dynamic 3D fluid surfaces with high accuracy.

1.5.2 Underwater Image Restoration

• We present refractive distortion guided network for restoring distortion-free underwater images by using distortion maps to guide network training. We first use a physically constrained convolutional network to estimate the distortion map from the refracted image. We then use a generative adversarial network guided by the distortion map to restore the sharp distortion-free image.

• We present a new fluid dataset that contains around 63K distorted refraction images, generated from 6354 unique reference patterns. Most of the reference patterns are selected from the Describable Textures Dataset(DTD) [20]. Except that, we include additional ~500 various texts images. We render 10 consecutive frames per wave. For each refraction image, we provide the ground truth distortion-free image (the reference pattern), the ground truth distortion map, and the ground truth height map of the wave.
• We evaluate our network on both real and synthetic underwater image datasets and show that it out-performs the state-of-the-art algorithms, especially in presence of large distortions. The perform the comparative experiments on three existing real dataset provided by Tian et. al [112], Li et. al [71] and Thapa et. al [111] (our previous real image dataset).

1.5.3 Non-rigid Image Distortion Removal

• We present a novel unsupervised network that jointly estimates the non-rigid distortions and recovers the latent distortion-free image. It works for both air and water distortions without much customization. It is fully unsupervised and does not need to be trained on a labeled dataset.

• Our network leverages the position encoding operator, such that even with fewer numbers of convolutional layers and trainable parameters, it can still generate high-quality images that preserve fine details.

• We propose a two-step optimization framework to guide the training of the unconstrained non-rigid distortion restoration model.

• Extensive experiments demonstrate that state-of-the-art performance can be achieved when applying the proposed grid-based rendering method on two inherently different tasks: atmospheric turbulence removal and imaging through water distortions.

1.6 Blueprint of the Dissertation

The remainder of this dissertation is organized as follows:

• Chapter 2 presents the novel deep neural network for dynamic fluid surface reconstruction. Here, we see the details of the network architecture, its loss functions. We also show the qualitative and quantitative results to verify its authenticity and limitless utilization.
• Chapter 3 presents the novel deep neural network for removing refractive distortions from underwater images. Here, we see how our proposed method estimates different types of fluid distortions in both synthetic and real scenes.

• Chapter 4 presents a novel unsupervised network for removing the refractive distortions of air and water medium. Here, we see the first attempt to predict the non-rigid motion based on grid deformations.

• Chapter 5 presents a combinational approach to simultaneously perform recovery of the latent distortion-free image as well as 3D reconstruction of the transparent and dynamic fluid surface.

• Chapter 6 concludes the paper and discusses the main directions for future work.
Chapter 2.
Deep Neural Network for Dynamic Fluid Surface Reconstruction

2.1 Introduction

Figure 2.1. Our dynamic fluid surface reconstruction scheme. Given a sequence of refraction images captured through the dynamic fluid and the original reference pattern, we develop a deep neural network to recover spatio-temporally consistent 3D fluid surfaces.

In this paper, we present a learning-based approach for reconstructing the 3D fluid surface from a single refraction image. Following the setting similar to the classical methods, we take refraction image of a reference pattern through the fluid from a top-down view. We design a deep neural network that takes the refraction image as input and generalize distortion features for 3D fluid surface reconstruction. In recent years, deep learning techniques

\footnote{This chapter previously appeared as S. Thapa, N. Li and J. Ye, "Dynamic Fluid Surface Reconstruction using Deep Neural Network," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 21-30. © 2020 IEEE. Reprinted with permission.}
have achieved great success in solving computer vision problems, including depth estimation [32, 33, 68, 103, 74], 3D reconstruction [19, 55, 120], object detection and recognition [41, 60, 68], . Although most networks assume Lambertian scenes as limited by existing datasets, there is a rising trend to apply deep neural networks for interpreting more complex scenes with reflection, refraction, and scattering. Stets [108] use a convolutional neural network to recover the shape of transparent refractive objects and show promising results. But both their network and dataset are not suitable for dynamic fluid surface reconstruction.

Specifically, our fluid surface reconstruction network (FSRN) consists of two sub-nets: 1) an encoder-decoder based convolutional neural network (FSRN-CNN) for per-frame depth and normal estimation and 2) a recurrent neural network (FSRN-RNN) for enforcing the temporal consistency across multiple frames. Our FSRN-CNN compares the refracted pattern image with the original pattern to learn features from distortion for depth and normal estimation. We explicitly account for the physics of refraction in our loss function for training. Our FSRN-RNN uses the convolutional long-short term memory (conLSTM) layers to learn temporal dependencies from previous frames, and refines the depth and normal estimation for the current frame to enforce spatio-temporal consistency. We train the two sub-nets separately to reduce the number of network parameters. Both are trained with per-pixel depth and normal losses as well as a depth-normal consistency loss. Since no existing dataset can serve our purpose for fluid surface reconstruction, we synthesize a large fluid image dataset with over 40,000 fluid surfaces for network training and validation. We use a set of fluid equations [21, 107, 110] derived from the Navier-Stokes for realistic fluid surface modeling. We implement a physics-based renderer that considers complex light transport to simulate images through refraction. Our dataset also includes the ground truth depth and normal maps of the fluid surfaces. We perform experiments on our synthetic dataset as well as real captured fluid images. Both qualitative and quantitative results show that our approach is highly accurate in recovering dynamic fluid surfaces.
2.2 Related Work

Classical image-based methods usually measure the refractive distortions of a known background pattern to recover fluid surfaces. We refer the readers to [52] for a comprehensive survey on refractive and reflective object reconstruction. Notably, Murase’s pioneering work [82] analyzes the optical flow between the distorted image and the original one to recover per-pixel normals for water surface. Tian and Narasimhan [113] develop a data-driven iterative algorithm to rectify the water distortion and recover water surface through spatial integration. Shan et al. [100] estimate the surface height map from refraction images using global optimization. As surface reconstruction from a single viewpoint suffers from the intractable depth-normal ambiguity [52, 80], most single image-based methods assume additional surface constraints such as planarity [4, 13, 35] and integrability [125, 130]. Morris and Kutulakos [80] first extend the classical multi-view geometry to refractive medium and recover the fluid surface using a stereo setup. Ding et al. [26] further adopt a $3 \times 3$ camera array for more robust feature tracking under distortion. Qian et al. [92] develop a global optimization framework to improve the accuracy of refractive stereo. Another class of computational imaging approaches directly acquire ray-ray correspondences using special optics [59, 130] and then triangulate the light rays for surface reconstruction. Being heavily dependent on the acquisition system, these classical methods usually use costly equipment that requires much effort to build and calibrate. In contrast, our approach allows for more flexible imaging setup and uses a learning-based algorithm for fluid surface reconstruction.

Deep learning techniques have achieved unprecedented success in numerous computer vision tasks including depth/normal estimation [32, 33, 68, 103, 74], 3D reconstruction[19, 55, 120] and object detection and recognition [41, 60, 70]. The encoder-decoder convolutional network architecture has proven effective in feature generalization for various applications. Most relevant networks are the ones for monocular depth/normal estimation. Eigen et al. [33] and Liu et al. [73] develop end-to-end trained convolutional networks for
depth estimation from a single image. Wang et al. [122] and Bansal et al. [6] use fully
c Connected convolutional networks with semantic labeling for single-image surface normal
estimation. Qi et al. [91] present a network for joint depth and normal estimation that
incorporates geometric constraints between depth and normal. However, all these networks
assume Lambertian scenes because they are trained on datasets that are mostly composed
of diffuse objects (e.g., NYU Depth [22] and KITTI [38]). They are, therefore, not applic-
able to recover fluid surfaces with reflective and refractive reflectance. Most recently, Li
et al. [71] present a network to un-distort the refractive image of an underwater scene.
Stets et al. [108] use convolutional network to recover the shape of transparent refractive
objects. But these networks are not suitable for fluid surface reconstruction due to limita-
tions of their datasets. In this work, we create a large physics-based fluid surface dataset
with ground truth depth and normal. In addition, our network use recurrent layers [78] to
capture the temporal dynamics of fluid flows.

2.3 Network Design

In this section, we present our fluid surface reconstruction network (FSRN). We first
introduce the setting of our fluid surface reconstruction problem, and then describe our
network structure and the generation of our physics-based fluid dataset.

2.3.1 Problem Definition

We represent the dynamic 3D fluid surfaces as a temporal sequence of the surface depths
\{z^t|t = 1, 2, \ldots\}, where t is the time instant and z^t = f^t(x, y) is the height field of the fluid
surface at t. As is shown in Fig. 2.2, given a reference pattern I_r placed underneath the
fluid surface at the z = 0, we can map I_r to a refraction image I^t as being distorted by the
refraction that occurs at the fluid surface z^t:

\[ I^t = \Phi(I_r, z^t). \]  \hspace{1cm} (2.1)

where \( \Phi \) is the mapping function that follows the physics of refraction (, the Snell’s law).
Figure 2.2. The setting of our fluid surface reconstruction problem. Given a refraction image \( I \) viewed from the top through the fluid flow and the original background pattern \( I_r \), we aim at recovering the fluid surface in form of depth and normal maps. Our network explicitly accounts for the physics of refraction in the training loss function.

Given a sequence of the refraction images \( \{I^t|t=1, 2, \ldots\} \) and the reference pattern \( I_r \), we aim to estimate the dynamic fluid surfaces \( \{z^t|t=1, 2, \ldots\} \). Practically, \( I_t \) can be captured by an orthographic camera that looks at the fluid surface from the top and \( I_r \) is assumed known in advance. In our network, we estimate both the depth map and the normal map of a fluid surface as they can be independently inferred from the refractive distortions. Since they are also geometrically correlated, the depth and normal estimations can be further refined with a consistency loss. Finally, we can generate 3D fluid surface meshes from our estimated depths and normals through Delaunay triangulation.

2.3.2 Network Architecture

Our fluid surface reconstruction network (FSRN) consists of two sub-nets: 1) an encoder-decoder based convolutional neural network (FSRN-CNN) for per-frame depth and normal estimation and 2) a recurrent neural network (FSRN-RNN) for enforcing the temporal consistency across multiple frames. Fig. 2.3 shows the workflow of our FSRN and Fig. 4.2 shows its architecture.
Figure 2.3. The workflow of our FSRN. The FSRN-CNN estimates depth and normal maps given a refraction image and the reference pattern. Its output is then structured into a temporal sequence and fed into the FSRN-RNN for refinement by enforcing the temporal consistency.

**FSRN-CNN.** Our CNN subnet takes in the refraction image $I^t$ and the reference pattern $I_r$ to estimate the depth map $D^t$ and normal map $N^t$ of the fluid surface at time $t$ (superscript $t$ indicates the time instance). It uses the encoder-decoder structure to generalize features from refractive distortion. The encoder is consisted of stacked convolutional layers with max-pooling layers. The decoder is made up of transpose convolutional layers with skip connections (see Fig. 4.2). Specifically, our decoder has two branches: one predicts normalized depth and normal maps ($D^t$ and $N^t$), and the other predicts the absolute ranges of depth and normal maps ($R^t$ and $R^t$). In order to generalize scale-invariant features, we normalize our depth and normal maps to the range of $0, 1$. These ranges are therefore critical to restore the actual scale of the fluid surface. To better exploit the geometric consistency between depth and normal, we use a common set of decoding layers for both depth and normal estimation. This subnet is end-to-end trained with loss functions described in Sec. 2.3.3. Table 2.1 provides the detailed network architecture of the FSRN-CNN.
Table 2.1. Detailed network architecture of FSRN-CNN.

<table>
<thead>
<tr>
<th>Input</th>
<th>Filters</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>18@2 x 2 x 6</td>
<td>128 x 128 x 6</td>
</tr>
<tr>
<td>Conv+Conv</td>
<td>36@2 x 2 x 18</td>
<td>64 x 64 x 36</td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>32 x 32 x 36</td>
</tr>
<tr>
<td>Conv+Conv</td>
<td>72@2 x 2 x 36</td>
<td>32 x 32 x 72</td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>16 x 16 x 72</td>
</tr>
<tr>
<td>Conv+Conv</td>
<td>144@2 x 2 x 72</td>
<td>8 x 8 x 144</td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>4 x 4 x 144</td>
</tr>
<tr>
<td>Deconv</td>
<td>72@4 x 4 x 144</td>
<td>8 x 8 x 72</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>36@6 x 6 x 72</td>
<td>16 x 16 x 36</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>18@4 x 4 x 36</td>
<td>32 x 32 x 18</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>9@4 x 4 x 18</td>
<td>64 x 64 x 9</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>4@3 x 3 x 9</td>
<td>128 x 128 x 4</td>
</tr>
<tr>
<td>Output1</td>
<td></td>
<td>128 x 128 x 4</td>
</tr>
<tr>
<td>Deconv</td>
<td>72@4 x 4 x 144</td>
<td>8 x 8 x 72</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>36@6 x 6 x 72</td>
<td>16 x 16 x 36</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>18@4 x 4 x 36</td>
<td>32 x 32 x 18</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>9@4 x 4 x 18</td>
<td>64 x 64 x 9</td>
</tr>
<tr>
<td>Concat+Deconv</td>
<td>1@3 x 3 x 9</td>
<td>128 x 128 x 1</td>
</tr>
<tr>
<td>Output2</td>
<td></td>
<td>128 x 128 x 1</td>
</tr>
</tbody>
</table>

**FSRN-RNN.** Our RNN subnet refines the depth and normal estimation by enforcing the temporal consistency. We concatenate multiple scaled depth and normal maps estimated by the FSRN-CNN as temporal sequences: \( \{D^t\}_{t = t, t - 1, t - 2, \ldots} \) and \( \{N^t\}_{t = t, t - 1, t - 2, \ldots} \). The temporal sequences of depth and normal maps are then used as input to feed into the FSRN-RNN. The output is refined depth and normal maps at the current time \( t \). We use convolutional long-short term memory (conLSTM) layers [104] to construct our recurrent network. The conLSTM layers transmit hidden states from previous time frames to learn the temporal dependencies. This subnet therefore enforces temporal consistency in our reconstruction as well as enhances the estimation accuracy.

The ablation study and real experiment results in Sec. 2.4 confirm the effectiveness of using the recurrent layers. This subnet is separately trained from the FSRN-CNN to reduce the
parameters. The loss functions are described in Sec. 2.3.3. Table 2.2 provides the detailed network architecture of the FSRN-RNN.

### 2.3.3 Loss Functions

**Depth Loss.** We use a per-pixel depth loss to compare our predicted depth map \(D\) with the ground truth one \(\hat{D}\). Similar to [32], we consider the L2-norm difference and scale-invariant difference (the first and second term in Eq. 2.2). The scale-invariant difference term panelize differences of opposite directions. It therefore preserves the shape of the surface regardless of the scale. In addition, we also consider a gradient term (the third term in Eq. 2.2) that takes the four-directional differences to favor smoother prediction. Let \(d(p) = D(p) - \hat{D}(p)\) be the per-pixel depth difference (where \(p \in [1, M]\) is the pixel index with \(M\) as the total number of pixels), our depth loss \(L_d\) is defined as

\[
L_d(D, \hat{D}) = \frac{1}{M} \sum_p d(p)^2 - \frac{1}{2M^2} \left( \sum_p d(p) \right)^2 + \frac{1}{M} \sum_p \sum_i \delta_i(p)^2 \quad (2.2)
\]

where \(i\) indicate the indices of four neighboring pixels of \(p\) and \(\delta_i(p) = d(i) - d(p)\) represents the four-directional difference of \(d(p)\).

**Normal Loss.** As we predict our depth and normal maps in the same decoder branch, the \(x\), \(y\), and \(z\) components of the normal map are estimated in three separate passes. Our normal loss function is similar to the depth loss except that the computation is extended to three channels. We also exclude the third smooth term because the normals tend to

---

**Table 2.2. Detailed network architecture of FSRN-RNN.**

<table>
<thead>
<tr>
<th>Input</th>
<th>Filters</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td>3 (\times) 128 (\times) 128 (\times) 4</td>
</tr>
<tr>
<td>convLSTM</td>
<td>4(\times)2 (\times) 4</td>
<td>3 (\times) 128 (\times) 128 (\times) 4</td>
</tr>
<tr>
<td>convLSTM</td>
<td>4(\times)2 (\times) 4</td>
<td>3 (\times) 128 (\times) 128 (\times) 4</td>
</tr>
<tr>
<td>convLSTM</td>
<td>4(\times)2 (\times) 4</td>
<td>128 (\times) 128 (\times) 4</td>
</tr>
<tr>
<td>Conv</td>
<td>4(\times)2 (\times) 4</td>
<td>128 (\times) 128 (\times) 4</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td>128 (\times) 128 (\times) 4</td>
</tr>
</tbody>
</table>
have more drastic changes than depth. Given the predicted normal map \( N \) and the ground truth one \( \hat{N} \), our normal loss \( L_n \) is defined as

\[
L_n(N, \hat{N}) = \frac{1}{M} \sum_p n(p)^2 - \frac{1}{2M^2}(\sum_p n(p))^2
\]

(2.3)

where \( n(p) = N(p) - \hat{N}(p) \) is the per-pixel difference.

**Depth-Normal Loss.** Since depth and normal are geometrically correlated, we use a depth-normal loss to enforce consistency between our depth and normal estimations. Specifically, given the predicted depth map \( D \), we convert it to its corresponding normal map \( (N_d) \) by taking the partial derivatives:

\[
N_d(p) = \left[ \frac{\partial D(p)}{\partial x}, \frac{\partial D(p)}{\partial y}, -1 \right]^\top.
\]

We then normalize the normal vectors to unit lengths and convert the ranges of their \( x \), \( y \), and \( z \) components to \( 0, 1 \)

converted normal map \( (N_d) \) with the ground truth normal map \( \hat{N} \). Our depth-normal loss \( L_{dn} \) is then defined as

\[
L_{dn}(N_d, \hat{N}) = \frac{1}{M} \sum_p n'(p)^2 - \frac{1}{2M^2}(\sum_p n'(p))^2
\]

(2.4)

where \( n'(p) = N_d(p) - \hat{N}(p) \) is the per-pixel difference.

**Refraction Loss.** We use a refraction loss to directly account for the physics of refraction that occurs at the fluid-air interface. We trace a refraction image using the predicted depth and normal maps and the original reference pattern. We then compare the traced image with the input refraction image to minimize their difference. Specifically, we assume all incident rays \( \vec{s}_1 \) to the fluid surface are \([0, 0, 1]^\top\) as we assume an orthographic camera with top-down view. Given a predicted fluid surface normal \( \vec{n} \), we can compute the refracted ray \( \vec{s}_2 \) by

\[
\vec{s}_2 = \frac{n_r}{n_i} \left[ \vec{n} \times (-\vec{n} \times \vec{s}_1) \right] - \vec{n} \sqrt{1 - \left( \frac{n_r}{n_i} \right)^2 (\vec{n} \times \vec{s}_1)^2}
\]

(2.5)

where \( n_i \) and \( n_r \) are the refractive indices of air and water.

We then use the predicted depth values to propagate \( \vec{s} \) and intersect with the reference
pattern. The colors of the intersection points are returned to form our predicted refraction image \((I)\). We then use the L2-norm difference to compare \(I\) with the ground truth refraction \(\hat{I}\), which is the input to our network. Our refraction loss \(L_r\) is defined as

\[
L_r(I, \hat{I}) = \frac{1}{M} \sum_p (\hat{I}(p) - I(p))^2
\]  

(2.6)

**Scale Loss.** As our CNN subnet also predicts the absolute ranges of depth and normal maps in order to restore them to the actual scale, we simply use the L2-norm difference to compare our predicted ranges \((R_D \text{ and } R_N)\) with the ground truth ones \((\hat{R}_D \text{ and } \hat{R}_N)\). Our ground truth ranges are obtained by taking the minimum and maximum values of the depth and normal maps \(^2\) \((\hat{R}_D = [\min(D), \max(D)])\). Our scale loss \(L_s\) is defined as

\[
L_s(R_{D,N}, \hat{R}_{D,N}) = \frac{1}{M} \sum_p (R_{D,N}(p) - \hat{R}_{D,N}(p))^2
\]  

(2.7)

**Total Losses.** Our two sub-nets are trained separately to reduce the number of network parameters. The total losses for FSRN-CNN \((L_{CNN})\) and FSRN-RNN \((L_{RNN})\) are combinations of the above described losses

\[
L_{CNN} = \alpha_1 L_d + \alpha_2 L_n + \alpha_3 L_{dn} + \alpha_4 L_r + \alpha_5 L_s
\]  

(2.8)

\[
L_{RNN} = \beta_1 L_d + \beta_2 L_n + \beta_3 L_{dn}
\]  

(2.9)

\(\alpha_1, ..., \alpha_5\) and \(\beta_1, \beta_2, \beta_3\) are weighted factors and they are separately tuned for each subnet. Notice that we only use the refraction loss in FSRN-CNN as this computation is expensive and it’s more efficient to apply it on a single frame rather than a temporal sequence. We also exclude the scale loss in FSRN-RNN because the input to this subnet has already been scaled to their actual ranges.

\(^2\)For the normal map, we treat the three channels separately but in the same manner.
2.3.4 Physics-based Fluid Dataset

It is challenging to acquire fluid dataset with ground truth surface depths and normals using physical devices. We resort to physics-based modeling and rendering to synthesize a large fluid dataset for our network training. We use fluid equations derived from the Navier-Stokes to model realistic fluid surfaces and implement a physics-based renderer to simulate refraction images.

![Sample images from our fluid dataset. From top to bottom, we show waves simulated by the shallow water equation, Grestner’s equation, Gaussian equation, and sinusoidal equation. The patterns used are checkboard, tiles, concrete, and perlin noise. Specifically, we use an Eularian mesh-based fluid simulation to model fluid surfaces. We use a variety of fluid equations derived from the Navier-Stoke to account for the versa-tility of natural fluid flows. The fluid equations we use include The shallow water equations [21], Grestner's equations [110], Gaussian equations, and sinusoidal equations. We choose these wave equations as they model different behaviors of fluid waves. The shallow water]
equations are a set of partial differential equations derived from the Navier-Stokes. They describe in-compressible property of fluid where the mass and linear momentum is conserved. The Grestner’s equations are widely used in computer graphics to simulate ocean waves. We use them to model fluid with relatively large volumes. The Gaussian equations are used for creating water ripples with damping effects. The sinusoidal equations are used to model linearly propagating waves. More details of these waves equations can be found in the supplementary material. We use weighted linear combination of these equations to simulate the 3D fluid surfaces that are used in our dataset.

To render refraction images, we implement a ray tracer that considers the refraction of light. We setup our scene following the configuration shown in Fig. 2.2, where the camera, 3D fluid surface and the reference pattern are center-aligned. We trace rays from an orthographic camera and use Eq. 2.5 to compute the refracted rays (where we assume the indices of refraction for air and fluid are 1 and 1.33). The refracted rays are then traced to the reference pattern to form the refraction image.

Our dataset contains over 45,000 refraction images (75 fluid sequences) with the ground truth depth and normal maps. We also use a variety of reference patterns to enrich our dataset, which include noise patterns (Perlin, Simplex, and Worley), checkerboards with different sizes, and miscellaneous textures (bricks, tiles, etc). Sample images from our dataset are shown in Fig. 3.8.

2.3.5 Wave Equations for Fluid Modeling

In this section, we provide details of the wave equations that we use for fluid modeling when creating the physics-based fluid dataset. We incorporate four types of wave equations: the shallow water wave equations, Grestner’s wave equations, Gaussian equations, and sinusoidal equations.

The shallow water wave equations are a set of partial differential equations derived from the Navier-Stokes [21]. They describe in-compressible property of fluid where the mass and linear momentum is conserved. Let \( z_{sh} \) be the fluid surface height, \((x, y)\) be the
Eulerian mesh grid and \( t \) be the time instant, the differential equations can be written as

\[
\begin{align*}
\frac{\partial (\rho z_{sh})}{\partial t} + \frac{\partial (\rho z_{sh}u)}{\partial x} + \frac{\partial (\rho z_{sh}u)}{\partial y} &= 0 \\
\frac{\partial (\rho z_{sh}u)}{\partial t} + \frac{\partial (\rho z_{sh}u^2 \frac{1}{2} \rho g z_{sh}^2)}{\partial x} + \frac{\partial (\rho z_{sh}uv)}{\partial y} &= 0 \\
\frac{\partial (\rho z_{sh}v)}{\partial t} + \frac{\partial (\rho z_{sh}uv)}{\partial x} + \frac{\partial (\rho z_{sh}v^2 \frac{1}{2} \rho g z_{sh}^2)}{\partial y} &= 0
\end{align*}
\]  
(2.10)

where \((u, v)\) is the 2D velocity, \( \rho \) is the fluid density and \( g \) is the gravity acceleration constant.

The Grestner’s wave equations are widely used in computer graphics to simulate ocean waves [110]. We use them to model fluid with relatively large volumes. In our implementation, we compute the Grestner’s equation with its Fast Fourier Transform (FFT) form. The FFT-based representation of the equation can be written as

\[
z_{gr} = \sum_m \sum_n \tilde{z}_{gr} \exp(j2\pi(mx + ny))
\]  
(2.11)

where \( \tilde{z}_{gr} \) is the Fourier amplitude, \( j \) is the imaginary unit, \( m \) and \( n \) are integers bounded by \([-M/2, M/2]\) and \([-N/2, N/2]\) (\( M \) and \( N \) are the dimensions of the mesh grid). We use the Phillips spectrum [110] as our height amplitude Fourier component (\( \tilde{z}_{gr} \)) that determines the structure of the fluid surface.

Gaussian equations are used for creating water ripples with damping effects. Let \( z_{ga} \) be the fluid surface height, the equation can be written as

\[
z_{ga} = A \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \sin(\omega t)
\]  
(2.12)

where \( A \) is the wave amplitude, \( \sigma \) is the damping factor, and \( \omega \) is the phase factor.

Sinusoidal wave equations provide good approximation for the fluid dynamics. We use them to model linearly propagating waves. Let \( z_{si} \) be the fluid surface height, the
equation can be written as

\[ z_{si} = \sum_i A_i \sin \frac{2\pi}{\lambda_i} (x + c_i t) \]  \hspace{1cm} (2.13)

where \( A_i \) is the wave amplitude, \( \lambda_i \) is the wavelength, and \( c_i \) is the phase constant with respect to time.

**The overall wave equation** for modeling the fluid surface is a weighted linear combination of Equation 2.10 - Equation 2.13 and is written as

\[ z = \zeta_1 z_{sh} + \zeta_2 z_{gr} + \zeta_3 z_{ga} + \zeta_4 z_{si} \]  \hspace{1cm} (2.14)

where \( \zeta_1, ..., 4 \) are weight coefficients. We are able to create various types wave that are physically realistic with Equation 2.14.

**2.4 Experiments**

In this section, we evaluate our approach through both synthetic and real experiments.

**2.4.1 Network Training**

We implement our FSRN in TensorFLow [1] with around 1.7 million trainable parameters. All computations are performed in a computer with Xeon E5-2620 CPU and two NVIDIA GTX 1080 Ti GPUs. We segment our fluid dataset into 40,000 training images, 5,000 validation images and 1000 testing images. We set the parameters \( \alpha_{1,5} = 0.2 \), \( \beta_{1,2} = 0.4 \), and \( \beta_3 = 0.2 \) for our total loss functions. It takes around 6 hours to train our network.

Our FSRN is trained in two steps. First, we train the FSRN-CNN with our fluid dataset. We process the training data by normalizing the input (refraction image, depth and normal maps) to the range \([0, 1]\) slice-by-slice and save their true scale ranges. We use the Adam optimizer to train the network. We use batch size 32 for both training and
validation. We initialize the learning rate as $10^{-3}$ and decrease it by half after 15 epochs. We train the network for 35 epochs till convergence. Second, we train the FSRN-RNN with a temporal sequence of re-scaled predictions from FSRN-CNN as input. Here we consider three consecutive frames. We use the Adam optimizer to train this network with a fixed learning rate of $10^{-3}$. The batch size is 32 for both training and validation. We train the network for 15 epoch till converge.

2.4.2 Experiments on Synthetic Data

We first evaluate our approach on our synthetic fluid dataset. Our validation set contains 5,000 refraction images (20 unique dynamic fluid videos) that doesn’t overlap with the training set. These data is rendered with various types of reference patterns. Our fluid surface reconstruction results are shown in Fig. 2.6. More dynamic fluid video results can be found in our supplementary material. We can see that our recovered fluid surfaces are highly accurate and well preserve the wave structure of the ground truth.

![Figure 2.6. Fluid surface reconstruction on synthetic data.](image-url)
Table 2.3. Quantitative comparison with existing methods on depth estimation. We highlight the best performance in **bold**.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Error metric</th>
<th>Accuracy metric</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>Abs Rel</td>
<td>$\rho&lt;1.25$</td>
<td>$\rho&lt;1.25^2$</td>
<td>$\rho&lt;1.25^3$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DenseDepth [2]</td>
<td>0.851</td>
<td>0.408</td>
<td>0.016</td>
<td>0.033</td>
<td>0.051</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefractNet [108]</td>
<td>0.303</td>
<td>0.274</td>
<td>0.226</td>
<td>0.422</td>
<td>0.584</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSRN-S</td>
<td>0.262</td>
<td>0.247</td>
<td>0.338</td>
<td>0.572</td>
<td>0.710</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSRN-CNN</td>
<td>0.128</td>
<td>0.105</td>
<td>0.557</td>
<td>0.803</td>
<td>0.896</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FSRN (Ours)</strong></td>
<td><strong>0.126</strong></td>
<td><strong>0.098</strong></td>
<td><strong>0.562</strong></td>
<td><strong>0.812</strong></td>
<td><strong>0.901</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.7. Visual comparison with existing methods on depth estimation. All depth maps are normalized to [0, 1] for fair comparison.

We also perform quantitative evaluation in comparison with existing methods. As there are not many networks designed for fluid surface reconstruction, we choose two networks to compare with: 1) the RefractNet by Stets [108], which is a CNN designed to reconstruct transparent refractive objects and 2) the DenseDepth by Alhashim and Wonka [2], which is a latest state-of-the-arts network for single-image depth estimation but designed for Lambertian scenes. As the DenseDepth doesn’t perform well in our task, we didn’t pick other Lambertian scene-based depth estimation network for comparison. We use five error metrics following [33] to evaluate our prediction: the root mean square error (RMSE), the absolute relative error (Abs Rel), and three threshold accuracies ($\rho < 1.25, 1.25^2, 1.25^3$). Formulas for computing these error metrics can be found in our supplementary material.
As both the RefractNet and DenseDepth take a single image as input, for fair comparison, we also implement a single-input variation of our network (FSRN-S) that only take the refraction image (without the reference pattern) by not considering the refraction loss. We also compare with the depth prediction directly obtained from FSRN-CNN (without using the FSRN-RNN). All networks are trained on our fluid dataset. The quantitative comparison results are shown in Table 4.3. We can see that our FSRN out-performs the existing methods in all error metrics. We also show the visual comparison of predicted depth map in Fig. 2.7. We can see that the Lambertian scene-based method (DenseDepth) is unable to produce meaningful prediction. Although the RefractNet can recover some ripple waves, their overall estimation is highly noisy and inaccurate. In contrast, our FSRN can estimate highly accurate depth for fluid surface. And even without using the reference pattern and recurrent layers, our prediction still out-performs the existing methods.

### 2.4.3 Ablation Study

We perform ablation study to demonstrate the effectiveness of loss functions. In particular, we create three variations of our network: 1) FSRN-CNN\(_1\) that only uses the
basic depth and normal losses; 2) FSRN-CNN$_2$ that adds the depth-normal loss; and 3) FSRN-CNN$_3$ that also adds the refraction loss. We also compare with the FSRN-CNN subnet without using the recurrent layers. FSRN is our full proposed network that uses both sub-nets with the complete set of loss functions. The quantitative comparison results for both depth and normal estimations are shown in Table 2.4. We can see that performance of our network gradually improves as we incorporate more loss functions. This indicates that our depth-normal loss, refraction loss, and the recurrent subnet are effective and help improve the accuracy of prediction. We refer the readers to our supplementary material for visual comparisons of our ablation study. Figure 2.8 shows the visual comparison of the different sub-nets.

![Visual comparisons](image)

**Figure 2.8.** Visual comparisons for the ablation study. Left: we show the input refraction image and the original reference pattern. Middle: the recovered depth and normal maps of our network variants in comparison with the ground truth ones. Right: RMSE error plot for the network variants. We can see that the errors (for both depth and normal) decrease, as we incorporate more components in the loss function as well as using the RNN subnet.

### 2.4.4 Experiments on Real Data

We also perform real experiment to evaluate our network. Our experimental setup is shown in Fig. 2.9. We use a water tank with size $12 \times 24 \times 18$ inches for wave simulation. Our reference pattern is placed at the bottom of the tank. We use a variety of patterns (Perlin noise, pool liners, river rocks, and sands) to test the robustness of our approach. We mount a machine vision camera (FLIR GS3-U3-32S4C-C) to the top to record videos of the water wave. As we assume orthographic camera model, we mount the camera high
(around 50cm to the tank bottom) and use a long focal length lens (50mm, Horizontal FoV 8°) to minimize the perspective effect. We also use a small aperture size (f/8) to extend the depth-of-field. We further calibrate the camera [139]. We use the camera intrinsic parameters to remove lens-related distortions and the extrinsic parameters to compensate camera rotations such that the image plane is frontal parallel to the fluid surface. We capture the dynamic fluid video with the reference pattern as background at a frame rate of 121fps and use fast shutter speed 1ms to reduce motion blur. We therefore place four LED light panels to surround the water tank in order to have sufficient light. We finally crop the regions with background pattern from our raw images and use them as input to our network.

![Figure 2.9. Our experimental setup for real data acquisition. Left: Sample reference patterns that we use for the real experiments; Right: We setup a camera on top of the fluid tank to capture refraction images of the reference pattern.](image)

Our real fluid surface reconstruction result is shown in Fig. 2.14. We can see our reconstruction is consistent with the refractive distortion. Please see the supplementary material for videos of recovered dynamic fluid surfaces. We also compare the 3D reconstruction results using FSRN-CNN and our full network FSRN (with the RNN subnet). The reconstruction results for three consecutive frames are shown in Fig. 2.11. We can see that the FSRN-CNN results obviously change more abruptly while our full network produces a smoother propagation. This indicates that our FSRN-RNN can effectively enforce the
temporal consistency in our reconstruction. We also perform re-rendering experiments to demonstrate the accuracy of our approach. We use our recovered fluid surface to re-render the distortion image as seen by the camera. We compare our re-rendered image with the actually captured refraction image (see Fig. 2.12). We can see that the pattern distortions are highly similar.

Figure 2.10. Reconstruction results on real data. Complete video of reconstruction can be found in the supplementary material.

Figure 2.11. Comparison between FSRN-CNN and our full network FSRN (with RNN).
2.4.5 Discussions

Our network is able to achieve good performance on both synthetic and real fluid data although it is trained on a synthetic dataset. This could be due to two reasons: 1) our physics-based fluid dataset preserve characteristics of natural fluid flows thanks to the diverse set of fluid equations we use to model the fluid surface and 2) we consider the physics of refraction in our loss function for more accurate reconstruction. However, the refraction loss requires to take the original reference pattern as input. This limits the application of network in outdoor fluid scenes. We can overcome this problem by incorporating a network similar to [71] that first estimates the undistorted pattern and then use it for computing the refraction loss.

In addition, we observe that our network produce more accurate prediction on noisy pattern (ex. sand and cement textures) than on regular patterns (ex. checkerboard and pool liners). This is because these noisy patterns contain more high-frequency components that better preserves the refractive distortion features.
Figure 2.14. Additional results on real data.
2.5 Conclusions

We have presented a deep neural network (FSRN) for dynamic fluid reconstruction from refraction images. We use a convolutional network for depth and normal estimation, and a recurrent network for enforcing the temporal consistency of the dynamic fluid. We consider the depth-normal consistency and the physics of refraction in our loss functions for training. We have also created a large fluid dataset using physics-based fluid modeling and rendering. Through both synthetic and real experiments, we have shown that our network can recover fluid surfaces with high accuracy. One future direction is to generalize our network to arbitrary background to eliminate the use of a reference pattern. We plan to further extend our network to handle more challenging fluid scenes with reflection and scattering. As there’s very few work on applying deep learning to non-Lambertian scene, we expect our network and dataset can serve as baseline for studying fluid scenes.
Figure 2.4. The overall architecture of our FSRN. Please refer to the supplementary material for more detailed parameters of our network.
Figure 2.13. Additional results on synthetic data.
Chapter 3.
Learning to Remove Refractive Distortions from Underwater Image

3.1 Introduction

Figure 3.1. We design a physics-based distortion guided network for underwater image correction. Our method predicts the distortion-free image, given three distorted water images.

In this paper, we present the distortion guided network (DG-Net) for restoring distortion-free underwater images. The key idea is to use a distortion map to guide network training. The distortion map models the pixel displacement caused by water refraction. As it reveals correspondences between the distorted image and the distortion-free one, we can use the refractive distortion to guide the network to make better predictions. We first use a convolutional neural network (CNN) to estimate the distortion map from the refracted image. Specifically, we design training losses that follow the physical model of refractive distortions, and exploit the temporal consistency among neighboring frames. We use three parallel CNNs to generalize features from each input; and then use recurrent layers to refine the CNN’s predicted distortion map. With the estimated distortion map, we can correct the slight refractive distortions in the input. Since large distortions are non-invertible and
resulting in many-to-one mapping, we then use a distortion guided generative adversarial network (GAN) to recover sharp distortion-free image. The distortion map is used to guide the training of both the generator and the discriminator of the GAN. Our network is trained on a synthetic refracted image dataset, with patterns that resemble the underwater environment.

We evaluate the DG-Net on our own synthetic dataset and several real-captured underwater image datasets [112, 111]. The results show that our method out-performs the state-of-the-arts[112, 88, 54, 71], especially in presence of large distortions. Compared with the model-based methods [112, 88], we don’t need long video sequence of a static underwater scene to achieve accurate reconstruction. Although we still take three images to exploit the temporal constraints, the images can be captured with the burst mode in a very short time interval. Our method can therefore be used for dynamic scenes. Compared with the learning-based methods [54, 71], our network requires fewer training data (around one tenth of the training set of [71]), but achieves better accuracy in presence of large distortions and generalizes well on arbitrary waves and underwater scenes.
Figure 3.3. The overall architecture of DG-Net. It consists of two subnets: a convolutional network for estimating the refractive distortions (Dis-Net) and a distortion-guided generative adversarial network for restoring the distortion-free image (DG-GAN). The generator and discriminator of DG-GAN are represented by $G$ and $D$, respectively. $F$ and $B$ denotes forward and backward mapping of images.

3.2 Related Work

The problem of recovering faithful underwater images is critical in underwater imaging scenarios. Early solutions [31, 67] take the mean/median of a distorted image sequence to approximate the latent distortion-free image. Although these methods work well for slight distortions, the mean image becomes blurry when the wave causes large distortions. Another popular class of methods rely on finding and stitching the “lucky patches” to recover the latent distortion-free image. Many solutions such as clustering [27, 28], manifold embedding [31], and Fourier-based averaging [124] are proposed to locate the “lucky patch” in the input sequence. The seminal work of Tian and Narasimhan [112, 113] presents a model-based tracking method to restore underwater images. Oreifej et al. [88] propose a two-step algorithm that first iteratively aligns the distorted images to the mean image and then denoises the estimation with low rank constraint. James et al. [57] track a set of salient feature points, and obtain the deformation field and the distortion-free image using a compressive sensing framework, by exploiting the Fourier-sparsity of the latent deformation fields. They further improve this method by using a Fourier decomposition of the so-called ‘displacement-trajectories’ derived from point-trajectories [58]. All these methods require a long sequence of distorted images as input and cannot work for single or few images.
They also require individual iteration for each sequence of images in-order to obtain the desired distortion-free image. Li et al. [71] propose a generative adversarial networks (GAN) to correct refractive distortions using a single image. In this work, our proposed network consider the physical model of refractive distortions, and use the distortion maps as training guidance. Our method can recover high quality distortion-free image with three input images.

**Estimating pixel displacement between images.** The problem of estimating pixel displacement has been extensive studies in motion/flow estimation. Most methods [48, 7] in this category consider rigid motion and estimate the displacement vectors through matched corresponding features. Recent trend is to use deep neural networks to tackle this problem. Dosoviskiey et al. [11, 81, 29] propose the FlowNet to estimate the shift between two consecutive images. Kanazawa et al. [62] propose the WarpNet to match invariant features between cross-category images. However, the refractive distortions caused by wavy water is highly non-rigid and it is difficult to find invariant features from the distorted images. Xue et al. [127] adopt the classical optical flow to estimate small refractive distortions caused by hot air or gas. In this work, we propose a physically-constrained convolutional network with recurrent layers to estimate large refractive distortions caused by wavy water.

**Image-to-image generation.** The generative adversarial networks (GANs) [43] have shown great success in solving image-to-image generation problems, such as image super-resolution [126, 10, 132, 123], denoising [136, 129, 15], deblurring [66, 135], inpainting [24, 121], etc. The key idea is to use an adversarial discriminator network (discriminator) to pit against the generative network (generator) and force the generator to produce more realistic images. Most existing GANs are trained with images of natural scenes [71, 9, 128, 23] or human faces [131, 115, 49] and require a large training set (with millions of images). Li et al. [71] modify the generator of [54] and use GAN to restore the distortion-free image. Without exploring any physical refractive constrains in their architecture, their network shows limited ability in predicting the true warp and restoring large refractive distortions.
In contrast, our GAN is trained with patterns that resemble the underwater environment. In addition, we use the refractive distortions to guide the training of both generator and discriminator. As result, our network requires fewer training data (~ 50k images), but achieves better accuracy.

3.3 Network Design

We consider the setting that a camera is looking at the underwater scene through the wavy water surface as shown in Fig. 3.1. The captured images therefore suffer from refractive distortions. Assume $J$ is the true image of the underwater scene unaffected by the water waves, our goal is to estimate a distortion-free image $\hat{J}$ that appears close to $J$ from the captured distorted images $I$. We propose the distortion guided network (DG-Net) to tackle this problem. Specifically, ground truth distortion maps are used to guide the training of our networks. The overall structure of our network is shown in Fig. 3.3. Our DG-Net has two subnets: a convolutional network for estimating the refractive distortions (Section 3.3.1) and a distortion-guided generative adversarial network for restoring the distortion-free image (Section 3.3.2). The two subnets are trained separately. Notice that although our network takes three sequential images $\{I^i\}_{i=1}^3$ as input, we only output one distortion-free image for the last frame ($I^3$). The first two frames are used as references for enforcing the temporal consistency of our distortion estimation. Unlike classical methods that require a video of static scene, our method can be used for moving scenes as the three sequential images can be captured with the burst mode in a very short time interval.

3.3.1 Distortion Estimation

We first use a distortion estimation network (Dis-Net) to predict the distortion map between the input distorted image and the latent distortion-free image. Our Dis-Net considers the physical model of refractive distortions and use temporal constraints to improve the estimation accuracy.
- **Refractive distortion model.**

  Given a distorted image $I$ and the true distortion-free image $J$, we define a distortion map $W = \{w_i\}_{i=1}^{M}$ (where $w_i \in \mathbb{R}^2$ is per-pixel distortion vector and $M$ is the total number of pixels) to represent the pixel displacement between $I$ and $J$ caused by the refraction of water-air interface. $w_i$ can then be written as:

  $$w_i = q_i - p_i$$  \hspace{1cm} (3.1)

  where $p_i \in \mathbb{R}^2$ is a pixel in $I$, and $q_i \in \mathbb{R}^2$ is a pixel in $J$. $q_i$ maps to $p_i$ through refraction.

  Since the refractive distortion is caused by the fluctuation of water surface, the amount of distortion (or pixel placement) is naturally related to the water surface height. By applying the first-order approximation of the Snell’s law, Tian and Narasimhan derive that the distortion vector $w_i$ has linear relationship with the gradient of surface height [112]. The mapping from the surface height map $H = \{h_i\}_{i=1}^{M}$ (where $h_i \in \mathbb{R}$ is a height value) to the distortion map $W$ can be written as:

  $$W = f(H) = \alpha \nabla H$$  \hspace{1cm} (3.2)

  where $\nabla = \left[ \frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right]$ is the gradient operator, and $\alpha = h_0(1 - \frac{1}{n})$ is a constant scalar determined by the average surface height $h_0$ and the refractive index $n$.

  The inverse mapping from $W$ to $H$ can then be found by integrating the distortion vectors:

  $$H = f^{-1}(W) = h_0 + \int \int_{x,y} \nabla H d_x d_y$$

  $$= \frac{\alpha \cdot n}{n - 1} + \int \int_{x,y} \frac{W}{\alpha} d_x d_y$$  \hspace{1cm} (3.3)

  As surface normals are related to the 2D height gradients, we can also derive the normal
Figure 3.4. Illustration of refractive distortion. $I$ and $J$ are distorted and distortion-free images respectively; $p_1$, $q_1$ and $p_2$, $q_2$ are two pairs of corresponding pixels; $h_0$ is the average water surface height, and $h_1$ is the height at the pixel $p_1$; $n_1$ and $n_2$ are normal vectors; $w_1$ and $w_2$ are distortion vectors.

map $N = \{n_i\}_{i=1}^M$ (where $n_i \in R^3$ is a normal vector) from the distortion map $W$ as:

$$n_i = \gamma_i \left[-\frac{w_i(x)}{\alpha}, -\frac{w_i(y)}{\alpha}, 1\right]$$

(3.4)

where $\gamma_i = 1/\|n_i\|$ is a normalization factor.

Given the surface height map $H$ and the distortion-free image $J$, the ground truth distortion map $W$ can be found by backward tracing rays from the image plane through the water surface to the underwater image $J$, as shown in Fig. 3.4. We use the ground truth distortion maps as well as physics-based losses derived from our refraction model to guide the training of the Dis-Net.

- **Network structure.**

The Dis-Net takes three distorted images $\{I^t\}_{t=1}^3$ as input, and output one distortion map prediction $\hat{W}$ for the last frame ($I^3$). The network consists of three concatenated convolutional neural networks (CNNs) followed by two recurrent layers (see Fig. 3.3).

Note that our network can be easily modified to take arbitrary number of images (by adding or reducing the CNN branches). We find that three images are sufficient to achieve decent performance even in presence of large distortions. Adding more input images results in more network parameters, but the performance gain is marginal.
Figure 3.5. The network architecture of a CNN branch of our Dis-Net.

The structure of a CNN branch is shown in Fig. 3.5. Each CNN estimate a distortion map from one distorted image. The encoder of our CNN is made up of standard stacked convolutional layers with max-pooling. The decoder uses variational refinement [29] to preserve fine details in the distortion map. Specifically, at each layer, we concatenate the transpose-convolved feature map, the corresponding feature map from the encoder, and an intermediate distortion map output by the current feature map. The intermediate distortion maps are compared with downsampled ground truth maps using our training losses.

The three distortion maps \( \{ \hat{W}t \}_{t=1}^{3} \) output from the CNNs are concatenated as a temporal sequence and fed into two stacked convolutional LSTM layers with batch normalization. The convolutional LSTM layers transmit hidden states from previous time frames to learn the temporal dependencies [105]. By exploiting the temporal consistency among the distortion maps, the prediction accuracy is further improved. The ablation study in Section 4.4.6 confirms the effectiveness of the recurrent layers.

- **Loss functions.**

  The Dis-Net takes ground truth distortion free image (\( J \)), distortion map (\( W \)) and surface height map (\( H \)) for training. We design loss functions following the refractive distortion model. Our loss functions consists of three terms: the distortion map loss, the refraction loss, and the consistency loss.

  The *distortion map loss* has three components. For each component, we use the scale-
invariant error function \([33]\) to measure the difference between two distortion maps:

\[
\varepsilon(W, W^*) = \frac{1}{M} \sum_{i=1}^{M} (w_i - w_i^*)^2 - \frac{1}{2M^2} \left( \sum_{i=1}^{M} (w_i - w_i^*) \right)^2
\] (3.5)

Intuitively, we compare the predicted distortion map \(\hat{W}\) with the ground truth one and calculate the error \(\varepsilon(W, \hat{W})\). Since the distortion map is directly related to water surface depth and normal, we consider two additional errors that are constrained by the physical models. Specifically, by applying Eq. 3.2 to \(H\), we can obtain another distortion map \(W_H\) converted from the ground truth height. We compare \(\hat{W}\) with \(W_H\) to enforce their consistency. By applying Eq. 3.3 and Eq. 3.4 to \(\hat{W}\), we can map our predicted distortion map to its corresponding height map \(\hat{H}\) and normal map \(\hat{N}\). We can then apply backward ray tracing and obtain a new distortion map \(W_{\hat{H}}\). We compare \(W_{\hat{H}}\) with the ground truth map \(W\). Since our converted height \(\hat{H}\) is accurate, the two map should be consistent.

In sum, our distortion map loss can be written as \(L_W = \alpha_1 \varepsilon(W, \hat{W}) + \alpha_2 \varepsilon(W_H, \hat{W}) + \alpha_3 \varepsilon(W, W_{\hat{H}})\), where \(\alpha_{1,2,3}\) are weighting factors.

The refraction loss minimizes the difference between the input image \(I\) and the distorted image \(I_{\hat{W}}\) traced with the height map \(\hat{H}\) mapped from \(\hat{W}\). We use the \(l^2\) norm as error metric: \(\varepsilon_{l^2}(I, I^*) = \frac{1}{M} \sum (I - I^*)^2\). Our refraction loss is therefore written as \(L_R = \varepsilon_{l^2}(I, I_{\hat{H}})\).

The consistency loss enforces consistent estimations from the three parallel CNNs. As three inputs \(\{I_t\}_{t=1}^{3}\) are captured in a short time interval, we assume their latent distortion-free images are the same. Specifically, we use the predicted distortion maps \(\{\hat{W}_{t}^*\}_{t=1}^{3}\) to undistort their corresponding inputs by applying Eq. 3.1 and obtain \(\{\hat{J}_{t}^{W*}\}_{t=1}^{3}\). We use the \(l^2\) error to compare pairwise difference among \(\{\hat{J}_{t}^{W*}\}_{t=1}^{3}\). The consistency loss is therefore written as \(L_C = \frac{1}{3} \sum_{t,s=1}^{3} \varepsilon_{l^2}(\hat{J}_{t}^{\hat{W}_t}, \hat{J}_{s}^{\hat{W}_s})\).

We combine \(L_W\), \(L_R\), and \(L_C\) to train the Dis-Net. The training is done end-to-end. The CNNs and the recurrent layers use different sets of weights for the losses.
Figure 3.6. Illustration of forward $\mathcal{F}$ and backward mapping $\mathcal{B}$. $I$ and $J$ are distorted and distortion-free images respectively, and $W$ is the distortion map. We can see that, in high distortion cases, backward mapping alone with the correct distortion map fails to get the accurate distortion-free image.

### 3.3.2 Image Restoration

Given the estimated distortion map $\hat{W}$, we propose a distortion guided adversarial network (DG-GAN) to estimate the distortion-free image $\hat{J}$. By directly applying $\hat{W}$ to undistort the input distorted image $I$, we can obtain an intermediate image $\hat{J}_W = \mathcal{B}(I, \hat{W})$, where $\mathcal{B}$ refers to backward mapping:

$$\mathcal{B}(I, W) = I(p - w) \quad (3.6)$$

We use this warped image $\hat{J}_W$ as input to the DG-GAN. Although $\hat{J}_W$ appear less distorted than $I$, some large distortions cannot be inverted as several pixels in $J$ may map to one pixel in $I$ through refraction as shown in Fig. 3.6.

Our DG-GAN has similar structure to the conditional GAN [54], but adopts distortion guided training losses. The generator $\mathcal{G}$ uses the “U-Net” as base architecture. It has 6 convolutional layers in the encoder and 6 deconvolutional layers with skip connections in the
Figure 3.7. Sample images of our synthetic underwater image dataset. From left to right, we show the ground truth (GT) distortion-free image, GT height map, GT distortion map, and the refraction image.

decoder. $G$’s goal is to produce distortion-free images $\hat{J}$ that cannot be distinguished from “real” by the discriminator. $G$ is trained with both the $l1$ and $l2$ losses that force its output to appear similar to the ground truth distortion-free image $J$. The $l1$ loss encourages less blurring and help to generate sharper image. In addition, we can apply the ground truth distortion map $W$ on $G$’s output to obtain a distorted image $\hat{I}_G = F(G(\hat{J}_W), W)$, where $F$ refers to forward mapping:

$$F(I, W) = I(p + w)$$  \hspace{1cm} (3.7)

If $G$’s output appears similar to $J$, then $\hat{I}_G$ should be consistent with the input distorted image $I$. The loss function for training $G$ is therefore written as:

$$L_G = \frac{1}{M} \left( \sum |G(\hat{J}_W) - J| + \sum (G(\hat{J}_W) - J)^2 + \sum (\hat{I}_G - I)^2 \right)$$  \hspace{1cm} (3.8)
The discriminator $D$ is adversarially trained to identify the “fake” hallucinated images from generator. Our $D$ is formed with 6 modules of the form convolution-BatchNorm-ReLu modules [53]. Besides learning a mapping from the input $\hat{J}_W$ to the distortion-free image $J$, the network also learns to predict whether the distortion constraint is satisfied. Specifically, we apply the ground truth distortion map $W$ to $\hat{J}_W$, the discriminator $D$ then favors predictions that appear closer to input distorted image $I$, instead of the forward mapping result $\hat{I}_G$ of $G$’s output. The objective function of our DG-GAN can be written as:

$$
\mathcal{L}_{GAN}(G, D) = E[\log D(\hat{J}_W, J)] + E[\log(1 - D(\hat{J}_W, G(\hat{J}_W)))] \\
+ E[\log D(F(\hat{J}_W, W), I)] \\
+ E[\log(1 - D(F(\hat{J}_W, W), \hat{I}_G))] 
$$  

(3.9)

$G$ tries to minimize Eq. 3.9 against the adversarial $D$ that tries to maximize it. The corrected distortion-free image is then optimized as $\hat{J} = \arg\min_{\hat{J}} \max_{\hat{I}} \mathcal{L}_{GAN}$. The ablation study in Section 4.4.6 confirms the effectiveness of our physics-based loss functions.

### 3.3.3 Network architecture

Here we provide the detailed architecture of our network. Our distortion guided network (DG-Net) consists of two sub-nets: 1) a convolutional network for estimating the refractive distortion from 3 consecutive refracted images (Dis-Net) and 2) a distortion-guided GAN for generating the undistorted underwater image (DG-GAN). Table 3.1 and Table 3.2 provide detailed network architecture of the two subnets. In the tables, Conv, Deconv and BN refer to convolution layers, transpose convolution layers, and batch normalization.

### 3.4 Experiments

In this section, we evaluate our DG-Net on both synthetic and real underwater image datasets. Specifically, we compare our method with competitive state-of-the-art methods,
Table 3.1. Detailed architecture of the Dis-Net.

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Input</th>
<th>Filters</th>
<th>Output Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td></td>
<td>128 × 128 × 3</td>
</tr>
<tr>
<td>Conv+Conv, ReLU</td>
<td>18@2 × 2 × 6</td>
<td>128 × 128 × 18</td>
<td></td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>64 × 64 × 18</td>
<td></td>
</tr>
<tr>
<td>Conv+Conv, ReLU</td>
<td>36@2 × 2 × 18</td>
<td>64 × 64 × 36</td>
<td></td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>32 × 32 × 36</td>
<td></td>
</tr>
<tr>
<td>Conv+Conv+Conv, ReLU</td>
<td>72@2 × 2 × 36</td>
<td>32 × 32 × 72</td>
<td></td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>16 × 16 × 72</td>
<td></td>
</tr>
<tr>
<td>Conv+Conv+Conv, ReLU</td>
<td>144@2 × 2 × 72</td>
<td>16 × 16 × 144</td>
<td></td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>8 × 8 × 144</td>
<td></td>
</tr>
<tr>
<td>Conv+Conv+Conv, ReLU</td>
<td>144@2 × 2 × 144</td>
<td>8 × 8 × 144</td>
<td></td>
</tr>
<tr>
<td>Maxpool</td>
<td>Stride = 2</td>
<td>4 × 4 × 144</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decoder</th>
<th>Deconv+Deconv</th>
<th>72@4 × 4 × 144</th>
<th>8 × 8 × 72</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output5</td>
<td></td>
<td>8 × 8 × 3</td>
<td></td>
</tr>
<tr>
<td>Concat+Deconv+Deconv</td>
<td>36@6 × 6 × 72</td>
<td>16 × 16 × 36</td>
<td></td>
</tr>
<tr>
<td>Output4</td>
<td></td>
<td>16 × 16 × 3</td>
<td></td>
</tr>
<tr>
<td>Concat+Deconv+Deconv</td>
<td>18@4 × 4 × 36</td>
<td>32 × 32 × 18</td>
<td></td>
</tr>
<tr>
<td>Output3</td>
<td></td>
<td>32 × 32 × 3</td>
<td></td>
</tr>
<tr>
<td>Concat+Deconv+Deconv</td>
<td>9@4 × 4 × 18</td>
<td>64 × 64 × 9</td>
<td></td>
</tr>
<tr>
<td>Output2</td>
<td></td>
<td>64 × 64 × 3</td>
<td></td>
</tr>
<tr>
<td>Concat+Deconv+Deconv</td>
<td>3@3 × 3 × 9</td>
<td>128 × 128 × 3</td>
<td></td>
</tr>
<tr>
<td>Output1</td>
<td></td>
<td>128 × 128 × 3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RNN</th>
<th>Input</th>
<th>3 × 128 × 128 × 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>convLSTM, BN</td>
<td>3@2 × 2 × 3</td>
<td>3 × 128 × 128 × 3</td>
</tr>
<tr>
<td>convLSTM, BN</td>
<td>3@2 × 2 × 3</td>
<td>128 × 128 × 3</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td>128 × 128 × 3</td>
</tr>
</tbody>
</table>
Table 3.2. Detailed architecture of the DG-GAN.

<table>
<thead>
<tr>
<th>Input</th>
<th>Filters</th>
<th>Output Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td></td>
<td>128 × 128 × 3</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, LeakyReLU</td>
<td>64@4 × 4</td>
<td>64 × 64 × 64</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, LeakyReLU</td>
<td>128@4 × 4</td>
<td>32 × 32 × 128</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, LeakyReLU</td>
<td>256@4 × 4</td>
<td>16 × 16 × 256</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, LeakyReLU</td>
<td>512@4 × 4</td>
<td>8 × 8 × 512</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, LeakyReLU</td>
<td>512@4 × 4</td>
<td>4 × 4 × 512</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, LeakyReLU</td>
<td>512@4 × 4</td>
<td>2 × 2 × 512</td>
</tr>
<tr>
<td>Conv, Stride = 2</td>
<td>512@4 × 4</td>
<td>1 × 1 × 512</td>
</tr>
<tr>
<td>Concat+Deconv, Stride = 2, BN, ReLU</td>
<td>512@4 × 4</td>
<td>2 × 2 × 512</td>
</tr>
<tr>
<td>Concat+Deconv, Stride = 2, BN, ReLU</td>
<td>512@4 × 4</td>
<td>4 × 4 × 512</td>
</tr>
<tr>
<td>Concat+Deconv, Stride = 2, BN, ReLU</td>
<td>512@4 × 4</td>
<td>8 × 8 × 512</td>
</tr>
<tr>
<td>Concat+Deconv, Stride = 2, BN, ReLU</td>
<td>256@4 × 4</td>
<td>16 × 16 × 256</td>
</tr>
<tr>
<td>Concat+Deconv, Stride = 2, BN, ReLU</td>
<td>128@4 × 4</td>
<td>32 × 32 × 128</td>
</tr>
<tr>
<td>Concat+Deconv, Stride = 2, BN, ReLU</td>
<td>64@4 × 4</td>
<td>64 × 64 × 64</td>
</tr>
<tr>
<td>Deconv, Stride = 2</td>
<td>3@4 × 4</td>
<td>128 × 128 × 3</td>
</tr>
<tr>
<td>Output_1</td>
<td></td>
<td>128 × 128 × 3</td>
</tr>
<tr>
<td>Discriminator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input_1</td>
<td></td>
<td>128 × 128 × 3</td>
</tr>
<tr>
<td>Input_2</td>
<td></td>
<td>128 × 128 × 3</td>
</tr>
<tr>
<td>Concat</td>
<td></td>
<td>128 × 128 × 6</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, ReLU</td>
<td>64@4 × 4</td>
<td>64 × 64 × 64</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, ReLU</td>
<td>128@4 × 4</td>
<td>32 × 32 × 128</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, ReLU</td>
<td>256@4 × 4</td>
<td>16 × 16 × 256</td>
</tr>
<tr>
<td>Conv, Stride = 2, BN, ReLU</td>
<td>512@4 × 4</td>
<td>8 × 8 × 512</td>
</tr>
<tr>
<td>Conv, BN, ReLU</td>
<td>512@4 × 4</td>
<td>8 × 8 × 512</td>
</tr>
<tr>
<td>Conv</td>
<td>1@4 × 4</td>
<td>8 × 8 × 512</td>
</tr>
<tr>
<td>Activation+Output</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
and perform ablation study on our network.

3.4.1 Network Training

- Data preparation.

Our DG-Net is trained on a large synthetic underwater dataset. We generate the dataset using physics-based modeling and rendering. Specifically, we use partial derivative equations derived from the Navier-Stokes to model the water waves. We consider waves with various heights and fluctuations to create distortions of different scales. Our dataset contains around 63k distorted refraction images, generated from 6354 unique reference pattern. Most of the reference patterns are selected from the Describable Textures Dataset (DTD) [20]. Except that, we include additional $\sim 500$ various texts images. We render 10 consecutive frames per wave. For each refraction image, we provide the ground truth distortion-free image (the reference pattern), the ground truth distortion map, and the ground truth height map of the wave. In Fig. 3.8, we show sample data from our dataset. The three consecutive frames are used as input to our algorithm.

We use the physics-based dynamic water wave simulation software presented in [111]. To show that our method is robust to different types of water wave fronts, we simulate three types of waves: ripple waves, ocean waves, and Gaussian waves. More details of these waves equations can be found in the supplementary material. Fig. 3.8 shows exemplary water distortion images and the corresponding distortion map and height map of the three types of waves. In the following, we describe the simulation equations used for each type of wave.

Ripple waves. We create water ripples with damping effects. Let $z_{ri}$ be the fluid surface height, the equation can be written as

$$z_{ri} = A \exp\left(\frac{-x^2 + y^2}{2\sigma^2}\right) \sin(\omega t)$$

(3.10)
where $A$ is the wave amplitude, $\sigma$ is the damping factor, and $\omega$ is the phase factor.

**Ocean waves.** The Grestner’s wave equations are widely used in computer graphics to simulate ocean waves [111, 110]. We use them to model fluid with relatively large volumes. In our implementation, we compute the Grestner’s equation with its Fast Fourier Transform (FFT) form. The FFT-based representation of the equation can be written as

$$z_{gr} = \sum_m \sum_n \tilde{z}_{gr} \exp(j2\pi(mx + ny)) \quad (3.11)$$

where $\tilde{z}_{gr}$ is the Fourier amplitude, $j$ is the imaginary unit, $m$ and $n$ are integers bounded by $[-M/2, M/2]$ and $[-N/2, N/2]$ ($M$ and $N$ are the dimensions of the mesh grid). We use the Phillips spectrum [110] as our height amplitude Fourier component ($\tilde{z}_{gr}$) that determines the structure of the fluid surface.

**Gaussian waves.** For Gaussian waves, we assume that the maximum water surface fluctuation is small compared to the height ($h_0$) of the fluid in the stable condition. This fluctuating water surface is governed by the wave equation:

$$z_{ga}(t + 1) = 2 \times z_{ga}(t) + c^2 \nabla - z_{ga}(t - 1) \quad (3.12)$$

Note that $z_{ga}(t = 0) = 0$ and $\nabla$ is a Laplacian operator related to $z_{ga}(t)$. Here, $c = \sqrt{gh_0}$ is the speed of the wave ($g$ is the gravity).

Fig. 3.8 shows exemplary water turbulence images of different types of waves. We also show the height maps and distortion maps, which are highly relevant to the water wave fronts.

The distortion-free underwater images are chosen from the Describable Textures Dataset (DTD) [20]. DTD contains a broad range of realistic texture images. We select a subset from the DTD whose appearance resembles the underwater scenes (for example, pool tiles,
sea plants, pebbles, etc.). In addition, we add \( \sim 500 \) various texts images to our set as underwater patterns. In sum, Our dataset contains \( \sim 63k \) distorted refraction images, generated from 6354 unique distortion-free images (or reference pattern). We keep 10 consecutive frames per wave. For each refraction image, we provide the ground truth distortion-free image, distortion map, and height map of the water surface. We divide our dataset as 70% for training (43,600), 15% for validation (9980), and 15% for testing (9960). Notice that all the waves and reference patterns are non-overlapping among the training set, validation set, and testing set. We plan to make our synthetic fluid dataset publicly available. Fig. 3.8 shows more samples of our dataset. We also show the height maps and distortion maps, which are highly relevant to the water wave fronts.

**Distortion levels.** In order to evaluate the robustness of our method with respect to the strength of distortion, we categorize our underwater images into seven distortion levels according to their distortion maps. Specifically, we quantify the distortion levels using the averaged magnitude of the distortion map. Given a distortion \( W = \{ w_i \}_{i=1}^M \) (where \( w_i \in \mathbb{R}^2 \) is per-pixel distortion vector and \( M \) is the total number of pixels), the distortion level \( d_l \) of its corresponding refraction image is

\[
d_l = \lceil \frac{1}{M} \sum_{i=1}^M \| w_i \|_2 \rceil
\]

where \( \lceil \cdot \rceil \) is the ceiling operator. In our dataset, \( d_l = 7 \) is the largest distortion level (\( d_l = 0 \) indicates distortion-free). We show exemplary images in selected distortion level in Fig. 3.17.

**Implementation details.**

We implement our network with TensorFlow [1]. The overall network (DG-Net) has around 50 million trainable parameters, which includes 3.1 million for the Dis-Net, 41 million for the generator of DG-GAN, and 6.9 million for the discriminator of DG-GAN. All computations are performed with a desktop computer with Xeon E5-2620 CPU and two NVIDIA GTX 1080 Ti GPUs.
Figure 3.8. More sample images of our synthetic underwater image dataset. From left to right, we show the ground truth (GT) distortion-free image, GT distortion map, GT height map, and three consecutive refraction images.
Figure 3.9. Visual comparison with [71] on the SynSet. We compare both the estimated distortion map and distortion-free image. We see that our predictions are robust to different types of wave fronts.

Figure 3.10. Visual comparison with the state-of-the-arts on the real captured TianSet [112]. Here Tian-61 and Oreifej-61 refer to using all 61 frames of a sequence as input to the methods [112] and [88].
The DG-Net is trained in two steps. We first train the distortion estimation network (Dis-Net) on the synthetic training set. We set the weights $\sim 0.55$, $\sim 0.25$, and $\sim 0.15$ for the distortion map loss $L_W$, refraction loss $L_R$, and consistency loss $L_C$ respectively. We use the Adam optimizer to train the network. We use batch size 64 for both training and validation with the learning rate of $10^{-4}$. We train the network with the loss functions described in Section 3.3.1 for 60 epochs until converge.

We then train the distortion guided generative adversarial network (DG-GAN) for restoring the distortion-free image. We use the estimate distortion map to backward map the input to an intermediate undistorted image, and we use it as input to the DG-GAN. We use the Adam optimizer to train DG-GAN with a fixed learning rate of $2 \times 10^{-4}$. We train the network with the loss functions described in Section 3.3.2 for around 400 epochs that suffices to produce good predictions.
Figure 3.12. Qualitative ablation on physics-based loss terms. We compare the predictions from different ablative sub-networks to the ground truth.

### 3.4.2 Comparison with the State-of-The-Arts

We compare our methods with the state-of-the-art underwater image restoration methods [112, 88, 54, 71, 57]. Specifically, Tian [112], and Oreifej [88] are two classical model-based approaches. Tian [112] use parametric models of distortion to restore the images. Oreifej [88] relies on per-frame registration with the mean image. James [57] is more recent compressed sensing (CS) solver for underwater image restoration. All these methods require a long input sequence to achieve good performance.

Isola [54] and Li [71] are recent learning-based methods for image generation/restoration. Isola [54] is a general-purpose pixel-to-pixel image generation network. It has good performance on style transfer, image coloring and inpainting. Li [71] is an adversarial network specifically for restoring refracted images, trained with $\sim$300k ImageNet images.

- **Testing sets.**

  We perform evaluations on four dataset: 1) **SynSet**: our synthetic dataset with 9960 testing images (generated with 996 different reference patterns); 2) **TianSet**: a real captured dataset by Tian [112], 3) **JamesSet**: we use the real captured dataset by James [57] (Cartoon, Elephant and Eye videos from [57] are for our testing purposes.) and 4) **ThapaSet**: a real captured dataset by Thapa [111]. TianSet, contains four real captured videos with refractive distortions. The four sequences use different reference patterns and each sequence has 61 frames. In our experiments, we also test Tian [112], Oreifej [88] and James [57] on shorter sequences with 10 frames. We take three real underwater scenes from ThapaSet. We also test [112], [88], and [71] on ThapaSet for further visual comparisons.
Regarding qualitative comparison on synthetic dataset (SynSet), we show the visual comparisons between our method and [112] in Fig. 3.9. We use the model of [112] trained

| Evaluation metrics. We use four standard image quality/similarity metrics for quantitative evaluation: 1) Peak Signal-to-Noise Ratio (PSNR) [51], 2) Structural Similarity Index (SSIM) [47], 3) Sum Squared Difference (SSD) [88], and 4) SSD in Gradient (SSDG) [88]. We compare the recovered distortion-free images with the ground truth ones to calculate these metrics.

- **Comparison results.**

Regarding qualitative comparison on synthetic dataset (SynSet), we show the visual comparisons between our method and [112] in Fig. 3.9. We use the model of [112] trained

---

<table>
<thead>
<tr>
<th>Table 3.3. Quantitative comparison with the state-of-the-arts.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>SynSet</td>
</tr>
<tr>
<td>Isola</td>
</tr>
<tr>
<td>Li</td>
</tr>
<tr>
<td><strong>DG-Net (Ours)</strong></td>
</tr>
<tr>
<td>Tian-61</td>
</tr>
<tr>
<td>Tian-10</td>
</tr>
<tr>
<td>Oreifej-61</td>
</tr>
<tr>
<td>Oreifej-10</td>
</tr>
<tr>
<td>Isola</td>
</tr>
<tr>
<td>Li</td>
</tr>
<tr>
<td>James-61</td>
</tr>
<tr>
<td>James-10</td>
</tr>
<tr>
<td><strong>DG-Net (Ours)</strong></td>
</tr>
<tr>
<td>Tian-61</td>
</tr>
<tr>
<td>Tian-10</td>
</tr>
<tr>
<td>Oreifej-61</td>
</tr>
<tr>
<td>Oreifej-10</td>
</tr>
<tr>
<td>Li</td>
</tr>
<tr>
<td>James-61</td>
</tr>
<tr>
<td>James-10</td>
</tr>
<tr>
<td><strong>DG-Net (Ours)</strong></td>
</tr>
<tr>
<td>Tian-61</td>
</tr>
<tr>
<td>Tian-10</td>
</tr>
<tr>
<td>Oreifej-61</td>
</tr>
<tr>
<td>Oreifej-10</td>
</tr>
<tr>
<td>Li</td>
</tr>
<tr>
<td>James-61</td>
</tr>
<tr>
<td>James-10</td>
</tr>
<tr>
<td><strong>DG-Net (Ours)</strong></td>
</tr>
</tbody>
</table>
Figure 3.13. More visual comparative results on real underwater image dataset provided by Tian [112], James [57], Thapa [111], and our real captured set. Note that all the comparison methods take 61 frames as input, i.e., Tian-61, Oreifej-61, and James-61 refer to using the 61 frames as inputs to the methods [112], [88], and [57], respectively. Our method takes only 3 images as its input.

on our dataset, as it yields better result than the pre-trained model (See Table 4.3). We compare both the estimated distortion map and the distortion-free images. We can see that our method achieves better accuracy on both as we account for the temporal consistency by three images as input. Notice that the model-based methods [112] and [88] cannot be applied on the SynSet as they need long input sequence.

Regarding qualitative comparison on real-capture sequences, we show visual comparisons between our method and the state-of-the-arts on a sequence from our real captured scene, the TianSet and ThapaSet in Fig. 3.10. Note that we used 10 frames to test on Tian [112], Oreifej [88], and James [57] methods (i.e., 10 frames, referred as Tian-10, Oreifej-10 and James-10 in Fig. 3.10). We see that the recovered images for [112, 88] are severely
downgraded. We can also see that our method, using only 3 distorted inputs, outperforms other methods in terms of distortion correction capacity and image sharpness. For qualitative comparison on [112, 88, 57] trained in 61 frames, please refer to our Fig. 3.13.

Table 4.3 shows quantitative comparisons of all methods on both the SynSet and real-captured ones (TianSet, JamesSet and ThapaSet). For fair comparison, Isola [54] is trained on our dataset. We can see that the general purpose GAN [54] doesn’t work well on correcting the refractive distortions. Our method achieves the best performance on most of the metrics. Except that Tian [112] has higher PSNR than our method on the TianSet. James-61 also has higher PSNR and SSIM on their proposed dataset (JamesSet). Our method ranks higher in other metrics, and also produces better visual results.

Table 4.3 shows quantitative comparisons of all methods on both the SynSet and real-captured ones (TianSet and ThapaSet). For fair comparison, [54] is trained on our dataset. We can see that the general purpose GAN [54] doesn’t work well on correcting the refractive distortions. Our method achieves the best performance on most of the metrics. Except that [112] has higher PSNR than our method on the TianSet. Our method ranks higher in other metrics, and also produces better visual results.

“In the wild” comparison. The existing methods James [57], Tian [112], Oreifej [88] are mostly tested on static and planar image patterns and require long sequence to restore the underlying underwater scene. However, in real scenarios, we have either camera shaking or dynamic moving objects in the water. To evaluate the robustness in handling more challenging indoor and outdoor scenarios, we captured a real turbulent dataset for comparison. Specifically, we used iPhone 11’s slow motion camera to capture two short videos of aquarium at frame rate 120 fps. One video contains mobile fishes, as shown in Fig. 3.14, and another video has moving aquatic plants in the water. Both videos are captured in a well-lit room. For outdoor scenarios, we used a lightweight DJI Mavic Mini drone to capture two videos (at 60 fps) of swimming pool with different patterns underneath, as shown in Fig. 3.2. The captured image sequences are therefore prone to camera shake.
Figure 3.14. Comparison on the “in the wild” dataset. Note that the top two are the images of the same fish and our recovery has the most consistent shape of the fish. The bottom is an image captured via our drone, which leads to severe camera shaking.
Figure 3.15. Visual results on Pool Scene taken with drone. The first column represents the distorted input image, the second column is our distortion-map estimation, the third column is the distortion-free image estimated by our Dis-Net and the last column is our final distortion-free image estimation from DG-GAN.

We compare with James [57], which achieved the best performance given a long sequence, and Li [71] that only need single image as input. As shown in Fig. 3.14, by taking only three distorted images as input, our method achieves the highest robustness in these challenging scenarios. Please find more results in pool scene in Fig. 3.15 and aquarium scene in Fig. 3.16.

3.4.3 Ablation Studies

- **Effect of the physical constrains.**

  We first perform ablative experiments on the physics-based loss terms described in Section 3.3.1 to show their effectiveness. Specifically, we compare our full network (DG-Net) with the Dis-Net (without the distortion-guided GAN), and three variants of the Dis-Net: 1) Dis-Net$_W$, which removes the last two terms of $L_W$ (notice that these terms are constrained by our physical model); 2) Dis-Net$_R$, which removes the refraction loss $L_R$ in Dis-Net; and 3) Dis-Net$_C$, which removes the consistency loss $L_C$ in Dis-Net. We train all these sub-networks on our SynSet for comparisons. The quantitative comparisons are shown in Table 3.4. We see that our final network DG-Net has the preferable high
Figure 3.16. Visual results on aquarium scene with one big fish (top two rows) and multiple tiny fishes (middle two rows) swimming, and aquatic plants (bottom two rows). We use the white arrows to point to the tiny fishes. The first column represents the distorted input image, the second column is our distortion-map estimation, the third column is the distortion-free image estimated by our Dis-Net and the last column is our final distortion-free image estimation from DG-GAN.
values in PSNR and SSIM metrics and low values in SSD and SSDG metrics. Qualitative comparisons of the un-distorted images are shown in Fig. 3.12. We can see that all loss terms contribute to improve our network performance.

- **Effect of the temporal constraint.**

To evaluate the effect of the temporal constraint, we create a single input version (DG-Net-S) of our full network by removing the recurrent layers and keeping only one CNN branch. Here we also compare with [71], as it takes a single image as input. We perform experiments on the SynSet. In addition to the recovered images, we also compare the estimated distortion map. We use the root mean square error (RMSE) and the absolute relative error (Abs Rel) to evaluate the distortion map. The quantitative comparison is shown in Table 3.5. We can see that our full network has the lowest error. We can also see that using recurrent layers to exploit the temporal consistency helps improve the performance.

- **Effect of refraction distortion constraint.**

To evaluate the effect of using distortion map as guidance, we create two variants of our network: 1) our network without distortion guidance (noted as ours w/o DG); and 2) our network without the DG-GAN (that leaves the Dis-Net alone).
Figure 3.17. Refraction images with different distortion levels. We show seven exemplary images of different distortions levels (Top) and the corrected output by our method (Bottom).

Figure 3.18. Comparisons with respect to the distortion levels.

We compare our full network with the two variants, as well as Li et al. [71] and Isola et al. [54]. Experiments are performed on refracted images with different distortion levels. We categorize the SynSet into seven distortion levels using Eqn. 3.13. The distortion level are quantified with the average magnitude of the distortion map, where level 0 indicates distortion-free, and level 7 indicates the strongest distortions. All the distortion levels of a sample image are shown in Fig. 3.17. Fig. 3.18 compares the PSNR of recovered images from all methods at 7 distortion levels. We can see that compared with the methods without distortion guidance, our method stays relatively robust for all distortion levels. Although certain distortions still persist when the input has high levels of distortions (see the example of level 7 in Fig. 3.17), our method still largely improve the image quality and make the scene discernible, which is of great importance to text scenes.
3.5 Conclusion

We have presented a physically constrained distortion guided network (DG-Net) for restoring distortion-free underwater images. By exploring the physics based constrains of refractive distortion and the water surface geometry, our network achieves outperforming performances on both synthetic and real datasets, compared with state-of-the-art algorithms, especially in handling large distorted image sequences. Our convolutional network (Dis-Net) exploits the physical model of refractive distortions and estimate the distortion map from the refracted image sequence. We adopt a generative adversarial network (DG-GAN) to restore a sharp distortion-free image by using the estimated distortion map to guide the network training.
Chapter 4.
Unsupervised Non-Rigid Image Distortion Removal via Grid Deformation

4.1 Introduction

Figure 4.1. We present an unsupervised network for predicting the distortion-free image and the distortion field, given distorted turbulent images. Our network works for both the air (first row) and water turbulence (second row).

In this paper, we develop an unsupervised deep neural network that is able to remove non-rigid distortions from both air and water turbulent images, as shown in Fig. 4.1. 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) [79, 109], 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.

The overall structure of our network is illustrated in Fig. 4.2. Our network consists
Figure 4.2. The overall architecture of our unsupervised non-rigid image distortion removal network. The network predicts the distortion-free image $J$, given an input distorted turbulent image $I$. $	ilde{I}$ and $\tilde{I}^G$ are two distorted images generated by our network. We use the pair-wise differences among $I$, $\tilde{I}$, and $\tilde{I}^G$ as the optimization losses.

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 [87] 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.

We perform extensive experiments on both simulated and real-captured air and water
turbulent images. We compare our method with the state-of-the-art methods that are specific to either the air or the water turbulence. We show that our method has better performance in correcting the geometric distortions for both types of turbulence.

4.2 Related Work

The problem of estimating pixel displacement has been extensively studied in motion/flow estimation. Most methods [48, 7] in this category consider rigid motion and estimate the displacement vectors through matched corresponding features. Recently, deep neural networks such as the FlowNet architectures [11, 29, 81] have achieved state-of-the-art in estimating the shift between two consecutive images. Kanazawa [62] proposed WarpNet to match invariant features between cross-category images.

However, the refractive distortions caused by wavy water or hot air is highly non-rigid and it is difficult to find invariant features from the distorted images. Most techniques for non-rigid image alignment can be classified into three broad categories: (1) feature matching techniques aim to match a set of sparse local features in the distorted image with those in the template [72, 90], (2) non-parametric and generative models for estimating deformation [97, 102], and (3) discriminative approaches to estimate templates directly [98, 140]. Tian and Narasimhan [113, 114] handle the problem of non-rigid deformation more generally using a globally optimal algorithm, and show applications for seeing through water and cloth deformation. Our paper presents a general framework for estimating and correcting non-rigid image distortions via grid deformation using an unsupervised neural network.

**Atmospheric turbulence removal.** To resolve the distortion and blur introduced by air turbulence, conventional turbulence restoration methods leverage optical flow [12, 83, 106], lucky regions fusion [118, 96, 36, 61] and blind deconvolution [39, 141] to recover images. Methods employing image registration with deformation estimation architecture can also resolve small movements of the camera and temporal variations due to atmospheric refraction [141, 45]. In a similar turbulence removal problem (not atmospheric), Xue [127]
adapt classical optical flow to estimate small refractive distortions caused by hot air or gas.

However, many of these methods have artifacts when reconstructing dynamic scenes with large amounts of motion. To counter this, methods have been introduced such as block matching [50], enforcing temporal consistency [85], using reference frames [18], and segmenting static background from moving objects [87, 44, 3]. One promising avenue of direction has been utilizing the physics of turbulence to create accurate forward models for image formation. Mao [77] achieve state-of-the-art performance by utilizing knowledge of anisoplanatic turbulence to create a physics-constrained prior for optimization.

In addition to classical methods, there have been some deep neural networks for air turbulence removal proposed. These are typically convolutional neural networks trained with synthetic or semi-synthetic turbulent data [37, 84, 14]. However, these supervised architectures have trouble with generalization outside of the training data (as do most supervised neural networks). In contrast, our neural network operates in an unsupervised fashion and does not require training data.

**Unsupervised learning for image restoration.** Recently, unsupervised or self-supervised learning using deep image priors [117] for image restoration tasks have enabled improved performance without the need for training data. In [117], 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 [75, 42, 94, 119].

Recently, analysis-by-synthesis techniques have demonstrated impressive capabilities for estimating visual information, particularly for inverse graphics problems [76, 69, 86, 133, 40, 5, 116]. Mildenhall [79] demonstrate how a multilayer perceptron (MLP) coupled with a special layer known as Fourier features [109] can estimate the 5D radiance field of a scene. More recently, [16, 17, 30, 95, 138] exploit the NeRF architecture to solve problems like view synthesis, texture completion from impartial 3D data, non-line-of-sight imaging recognition, etc. In our paper, we leverage Fourier features operator to help perform
analysis-by-synthesis for our deformed images.

4.3 Network Design

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 \( G_k^{\theta} \) 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 \( G_k = G_k^{\theta}(G_U) \). The image generator is a neural network acting as a parametric function \( \tilde{I} = I_{\phi}(G) \) that maps a grid \( G \) to an image \( \tilde{I} \). When the grid \( G_k \) from the grid deformer is used as input, \( I_{\phi} \) maps its parameters \( \phi \) to a distorted color image \( \tilde{I}_k \), which is compared to the corresponding image frame \( I_k \). At the same time, feeding a uniform grid \( G_U \) to the network \( I_{\phi} \), we can expect \( I_{\phi} \) map \( \phi \) to a distortion-free image \( J \), as shown in Fig. 4.2. We also use the predicted distorted grids \( \{G_1, ..., G_K\} \) to directly resample \( J \) and obtain another set of distorted images \( \{\tilde{J}_1^G, ..., \tilde{J}_K^G\} \) 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 \( I \), \( \tilde{I} \), and \( \tilde{J}^G \). To properly estimate sharp image details in the image generator, we leverage the latest positional encoding technique in [79, 109] to preserve fine-details in our recovered latent image, without the need for extra convolutional layers with many parameters, which we describe in Section 4.3.1. 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 4.3.1.

4.3.1 Network Structure

The overall structure of our non-rigid distortion removal network is shown in Fig. 4.2. Our network has two main components: the grid deformer and the image generator. Table 4.1 provides detailed network architecture of the two subnets. In the tables, Conv, BN, ReLU refer to convolution layers, batch normalization and Rectified Linear Unit; $\gamma$ refers to the GRFF position encoding component.

**Grid deformer $G_\theta$** 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 $I_k \in \mathbb{R}^{3\times H \times W}$, $G_k = G_\theta^k(G_U)$, where $\theta$ is the set of trainable network parameters. $G_\theta$ 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. Note that, we train a separate $G_\theta^k$ 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, , 30 fps [99, 71, 112, 88]. Using a single network to predict all

<table>
<thead>
<tr>
<th>Table 4.1. Architecture of grid deformer $G_\theta$ and the image generator $I_\phi$.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>$G_\theta$</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>$I_\phi$</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
these random distortion fields is challenging without empirical guidance from ground truth labels and strong temporal consistency constraints. Secondly, the network structure of $G_\theta$ is simple and has few parameters, and thus we can jointly optimize $\{G_k^\theta | k = 1, \ldots K\}$ with low memory consumption for GPU implementation. Please find a more detailed discussion in Section 4.4.6.

**Image generator** $I_\phi$ renders a color image $\tilde{I} \in R^{3 \times H \times W}$ when given a grid input $G \in \{G_1, \ldots, G_k, G_U\}$: $\tilde{I} = I_\phi(G)$. If the input grid is a deformed grid $G_k$, $I_\phi$ returns an image $\tilde{I}_k$ 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 $J \in R^{3 \times H \times W}$. $I_\phi$ share a similar network architecture with $G_\theta$. Since the output of $I_\phi$ is a color image, we apply a nonlinear Sigmoid activation function to the output layer. Please find more details about the structure of $G_\theta$ and $I_\phi$ in our supplementary material.
Position encoding via Fourier features

As pointed out by [93], 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 [109, 79]. 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 $I_\phi$. Let $v = (x, y)$ be a coordinate from the input grid. Its GRFF is computed as $\gamma(v) = [\cos(2\pi\kappa B v), \sin(2\pi\kappa B v)]$, where $\cos$ and $\sin$ are performed element-wise, $\kappa$ is a bandwidth-related scale factor, and $B \in \mathbb{R}^{128 \times 2}$ 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 $\gamma(v) \in \mathbb{R}^{256 \times H \times W}$.

It is worth noting that the choice of $\kappa$ in the image generator is pertinent to our network’s performance. In general, large $\kappa$ tends to have the network converge fast and very likely to end up at a local minimum. In this paper, we empirically pick $\kappa = 8$. We discuss the effect of GRFF in an ablative study in Section 4.4.6.

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 $G_\theta$ and $I_\phi$ 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 $\theta$ and $\phi$ 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 [87]. 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:

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

(4.1)

where $|\cdot|$ represents the absolute differences (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 $\theta'$ and $\phi'$ as the initialized weights.

As illustrated in Fig. 4.4, 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 4.4.6.

**Iterative refinement.** After our initialization step, we set out to learn the underlying distortion-free image through the following optimization model:
Figure 4.4. 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).
\[
\min_{\theta, \phi} \sum_k |\tilde{I}_k - I_k| + R(I_k), \text{ s.t. } \theta^0 = \theta', \ \phi^0 = \phi',
\]  

(4.2)

where \( \tilde{I}_k = \mathcal{I}_\phi(\gamma(G_k)) \) is the estimated distorted image, \( G_k = \mathcal{G}_\theta^k(G_U) \) is the deformed grid, \( R(I_k) \) is a regularizer, \( \theta^0 \) and \( \phi^0 \) are the initial weights of the network. We use \( R(I_k) \) to strengthen the interconnection between the predicted distortion-free image \( J = \mathcal{I}_\phi(\gamma(G_U)) \) and deformed grids \( \{G_k\} \):

\[
R(I_k) = |\tilde{J}_k^G - I_k| + |\tilde{J}_k^G - \tilde{I}_k|, 
\]  

(4.3)

where \( \tilde{J}_k^G \) is a resampled distorted image by grid sampling the deformed grid \( G_k \) on the recovered latent image \( J \), as shown in Fig. 4.3. We iteratively update the \( J \) using Eqn. 4.2 until networks converge.

4.4 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).

4.4.1 Experimental setup

Implementation details. Our network was implemented in Pytorch [89] 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. 4.4. We empirically pick $\kappa = 8$ as the bandwidth-related factor of the Fourier feature mapping operator for all experiments.

**Memory consumption.** 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 [71], our network restores comparable high-frequency details in the predicted image with less memory footprint.

### 4.4.2 Simulation Details

**Air turbulence simulation** We use the physics-based simulation software presented in [99] to render images affected by the air turbulence. Fig. 4.5 illustrates the simulation setup and the respective parameters used in the simulator. Numerical values of the parameters are given in Table 4.2.

![Air turbulence simulation setup](image)

Figure 4.5. 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 use the refractive image constant ($C_n^2$) to control the strength of the air
The simulator use the refractive image constant \( C \) 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. 4.6 shows exemplary images of the three turbulent strengths.

**Table 4.2. Air turbulence simulation parameters.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path length ( L )</td>
<td>2km</td>
</tr>
<tr>
<td>Height ( h )</td>
<td>4m</td>
</tr>
<tr>
<td>Aperture Diameter ( D )</td>
<td>0.08m</td>
</tr>
<tr>
<td>Focal Length ( d )</td>
<td>0.3m</td>
</tr>
<tr>
<td>Wavelength ( \lambda )</td>
<td>550nm</td>
</tr>
<tr>
<td>Turbulence Strength</td>
<td></td>
</tr>
<tr>
<td>Weak</td>
<td>( 1 \times 10^{-14} \text{ m}^{-2/3} )</td>
</tr>
<tr>
<td>Medium</td>
<td>( 1 \times 10^{-13} \text{ m}^{-2/3} )</td>
</tr>
<tr>
<td>Strong</td>
<td>( 1 \times 10^{-12} \text{ m}^{-2/3} )</td>
</tr>
</tbody>
</table>

![Fig. 4.6. 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) [39]).](image)

The simulator use 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. 4.6 shows exemplary images of the three turbulent strengths.

**Water turbulence simulation** We use the physics-based dynamic water wave simulation software presented in [111]. To show that our method is robust to different types of water turbulence, we simulate three types of waves: ripple waves, ocean waves, and Gaussian waves. In the following, we describe the simulation equations used for each type of wave.

**Ripple waves.** We create water ripples with damping effects. Let \( z_{ri} \) be the fluid surface height, the equation can be written as
\[ z_{ri} = A \exp(-\frac{x^2 + y^2}{2\sigma^2}) \sin(\omega t) \] (4.4)

where \( A \) is the wave amplitude, \( \sigma \) is the damping factor, and \( \omega \) is the phase factor.

**Ocean waves.** The Grestner’s wave equations are are widely used in computer graphics to simulate ocean waves [111, 110]. We use them to model fluid with relatively large volumes. In our implementation, we compute the Grestner’s equation with its Fast Fourier Transform (FFT) form. The FFT-based representation of the equation can be written as

\[ z_{gr} = \sum_m \sum_n \tilde{z}_{gr} \exp(j2\pi(mx + ny)) \] (4.5)

where \( \tilde{z}_{gr} \) is the Fourier amplitude, \( j \) is the imaginary unit, \( m \) and \( n \) are integers bounded by \([-M/2, M/2]\) and \([-N/2, N/2]\) (\( M \) and \( N \) are the dimensions of the mesh grid). We use the Phillips spectrum [110] as our height amplitude Fourier component (\( \tilde{z}_{gr} \)) that determines the structure of the fluid surface.

**Gaussian waves.** For Gaussian waves, we assume that the maximum water surface fluctuation is small compared to the height (\( h_0 \)) of the fluid in the stable condition. This fluctuating water surface is governed by the wave equation:

\[ z_{ga}(t + 1) = 2 \times z_{ga}(t) + c^2 \nabla - z_{ga}(t - 1) \] (4.6)

Note that \( z_{ga}(t = 0) = 0 \) and \( \nabla \) is a Laplacian operator related to \( z_{ga}(t) \). Here, \( c = \sqrt{gh_0} \) is the speed of the wave (\( g \) is the gravity).

Fig. 4.7 shows exemplary water turbulence images of the three types of waves. We also show the distortion fields, which are highly relevant to the water wavefronts.
Figure 4.7. Exemplary turbulence images of the three types of water waves (ripple, ocean, and Gaussian). We also show their distortion fields and the distortion-free image.

4.4.3 Evaluation on air turbulence

For the air turbulence, we compare with the following state-of-the-art methods: CLEAR [3] (source code provided), Oreifej [87], Zhang [134], Gao [37], and Mao [77]. [3, 134, 77] are physics-based approaches that use complex turbulence models. [37] 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 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. More details on the simulation parameters can be found in our supplementary material. The quantitative comparison results with respect to various turbulence levels are reported in
Figure 4.8. 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.

Table 4.3. Quantitative comparison on air turbulence data with various strengths.

<table>
<thead>
<tr>
<th>Strength</th>
<th>Metrics</th>
<th>Average</th>
<th>Our init.</th>
<th>[3]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weak</td>
<td>PSNR↑</td>
<td>25.10</td>
<td><strong>25.20</strong></td>
<td>18.31</td>
<td>24.29</td>
</tr>
<tr>
<td></td>
<td>SSIM↑</td>
<td>0.941</td>
<td>0.95</td>
<td>0.856</td>
<td><strong>0.984</strong></td>
</tr>
<tr>
<td>Medium</td>
<td>PSNR↑</td>
<td>19.48</td>
<td>19.85</td>
<td>14.09</td>
<td><strong>20.70</strong></td>
</tr>
<tr>
<td></td>
<td>SSIM↑</td>
<td>0.774</td>
<td>0.804</td>
<td>0.561</td>
<td><strong>0.904</strong></td>
</tr>
<tr>
<td>Strong</td>
<td>PSNR↑</td>
<td>17.08</td>
<td>17.12</td>
<td>12.51</td>
<td><strong>17.40</strong></td>
</tr>
<tr>
<td></td>
<td>SSIM↑</td>
<td>0.632</td>
<td>0.667</td>
<td>0.433</td>
<td><strong>0.799</strong></td>
</tr>
</tbody>
</table>

Table 4.3. 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 [46] 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* in Fig. 4.8. As we don’t have access to the codes of several methods [3, 134, 37], we di-
Figure 4.9. Visual comparison results on our real captured hot-air turbulence and long-range atmospheric turbulence images.

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

We show the qualitative comparison on our real captured data in Fig. 4.9. Here we only compare to the methods that we have access to their codes or the authors have provided us their results. Please refer to our supplementary materials for the video results on these sequences.

4.4.4 Evaluation on water turbulence

For the water turbulence, we compare our methods with the following state-of-the-arts: Tian [112] and Oreifej [88] are physics-based method. Li [71] is a learning-based method. All provided the source codes.

We perform experiments on two water turbulent image datasets: [111] and [71]. Thapa [111] 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
Figure 4.10. Visual comparisons on real water turbulence images provided by Li [71], which proposed a supervised GAN model to restore water turbulence.

different types of waves. [71] 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. 4.10. We can see that our method outperforms all the states-of-the-arts. To further validate the robustness, 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 [111]. The ocean waves are the most challenging, as they are more random and have more high-frequency turbulence components. As shown in Table 4.4, our method ranks higher on the Ripple and Ocean waves. Although [71] achieves higher PSNR/SSIM scores on the Gaussian wave, their results appear blurrier than ours (see visual comparisons in the supplementary material). Further, [71] requires training on ~320K images.

4.4.5 More Results on Synthetic and Real Data

- Air turbulence results

We show additional visual results on simulated air turbulence in Fig. 4.11. We compare on simulated air turbulence of three strength levels: weak, medium and strong. We compare with the state-of-the-art method CLEAR [3], 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 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. 4.12. We compare on simulated water turbulence of three types: ripple, ocean and Gaussian. We compare with the state-of-the-art methods Tian [112], Oreifej [88], and Li [71].

We show more results on the real captured water turbulence scenes in the supplementary video. The real captured turbulent videos are provided by [111]. 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 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.

<table>
<thead>
<tr>
<th>Types</th>
<th>Metrics</th>
<th>[112]</th>
<th>[88]</th>
<th>[71]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple</td>
<td>PSNR↑</td>
<td>20.40</td>
<td>21.24</td>
<td>20.70</td>
<td>23.63</td>
</tr>
<tr>
<td></td>
<td>SSIM↑</td>
<td>0.878</td>
<td>0.902</td>
<td>0.882</td>
<td>0.970</td>
</tr>
<tr>
<td>Ocean</td>
<td>PSNR↑</td>
<td>20.93</td>
<td>21.13</td>
<td>21.32</td>
<td>22.32</td>
</tr>
<tr>
<td></td>
<td>SSIM↑</td>
<td>0.891</td>
<td>0.901</td>
<td>0.833</td>
<td>0.964</td>
</tr>
<tr>
<td>Gaussian</td>
<td>PSNR↑</td>
<td>17.61</td>
<td>17.40</td>
<td>17.40</td>
<td>18.67</td>
</tr>
<tr>
<td></td>
<td>SSIM↑</td>
<td>0.787</td>
<td>0.789</td>
<td>0.789</td>
<td>0.833</td>
</tr>
</tbody>
</table>

84
Figure 4.11. Qualitative comparison on simulated air turbulence images with various strengths (weak, medium, and strong).

Figure 4.12. Qualitative comparison on simulated water turbulence images with various types of waves (ripple, ocean, and Gaussian).
Table 4.5. Comparison of the performance on the restoration ability of $G_\theta$.

<table>
<thead>
<tr>
<th>$G_\theta$</th>
<th>10 subnets</th>
<th>1 network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Conv2</td>
<td>Conv4</td>
</tr>
<tr>
<td>Total params</td>
<td>0.02M</td>
<td>1.33M</td>
</tr>
<tr>
<td>PSNR↑</td>
<td>19.23</td>
<td><strong>20.48</strong></td>
</tr>
<tr>
<td>SSIM↑</td>
<td>0.775</td>
<td><strong>0.790</strong></td>
</tr>
</tbody>
</table>

### 4.4.6 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 [99] to generate 100 different turbulence fields with controllable turbulence strength $C^2$ that are applied to a clear image to generate distorted image sequences.

- **Network structures of $G_\theta$.**

  For a fair comparison of the capability of different structures in encoding the deformed grid, we replace $G^k_\theta$ with several different CNNs, as shown in Table 4.5. 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 $G^k_\theta$. We also compare with the architecture that simply use a deep Autoencoder CNN (DAE) with skip connections [94] 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 4.6, we show the average PSNR/S-SIM and total running time (2,000 epochs) for 2, 5, 10, 15 and 20 frames. Please find the visual comparison results in our supplementary materials. Increasing the input number does benefit our restoration task, but we sacrifice time efficiency in order to do so. Since there are

86
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. 4.13, 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 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 \( \kappa \) in our supplementary materials.

- **Effect of initialization step.**
  To evaluate the effects of the network initialization, we created four variants of the network for comparison: 1) \( G_{\text{no init}} + \mathcal{I} \), that removes the initialization step of grid deformer \( G \); 2) \( G + \mathcal{I}_{\text{no init}} \), that removes the initialization step of image generator \( \mathcal{I} \); 3) \( \text{No init} \), that has no initialization step at all; and 4) \( G' + \mathcal{I}' \), that adds the initialization losses to the iterative refinement step. As shown in Fig. 4.13, taking out the initialization step from either the \( G_\theta \) 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 4.3.1.
Figure 4.13. Ablation study on different variants of our proposed network. We show the PSNR and SSIM vs. the number of iteration curves for comparison.

- **Effect of number of input images**

  In the main paper, we show the quantitative comparison results of different number of input images. Here Fig. 4.15 shows the visual comparison results. 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 $\kappa$ for Fourier feature mapping. We show the quantitative comparison results (average PSNR/SSIM) in Table 4.7, and qualitative comparison results on distortion-free image and distortion field prediction in Fig. 4.15.

Table 4.7. Quantitative comparison on varied Fourier feature mapping parameters $\kappa$. Red and Blue refer to the top and second best performance respectively.

<table>
<thead>
<tr>
<th>$\kappa$</th>
<th>0.1</th>
<th>1</th>
<th>8</th>
<th>10</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>19.38</td>
<td>19.29</td>
<td><strong>20.28</strong></td>
<td>20.01</td>
<td>16.24</td>
<td>13.89</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.627</td>
<td>0.800</td>
<td><strong>0.796</strong></td>
<td>0.754</td>
<td>0.373</td>
<td>0.372</td>
</tr>
</tbody>
</table>
4.5 Conclusions

We have presented an unsupervised non-rigid image distortion removal network via grid-deformation given a short sequence of turbulent images. Our network architecture can jointly estimate the sharp latent image as well as the non-rigid distortion. Our network does not require any ground truth turbulence models as guidance, and thus can be generalized to handle most non-rigid distortions, including both air and fluid turbulence cases. We plan to open source our code and data for reproducible research to the community.

Limitations and future directions. As our method does not have any physics-based constraints and it takes only 10 frames as input, a good initialization is critical for our algorithm to reach optimizing results. Further, our method does not solve cases where there is large motion in the scene in addition to turbulence distortions. For future directions, we hope to tackle these issues as well as make our method applicable to more general non-rigid distortions.
Chapter 5.
A Combined Approach to Solving Fluid Type Phenomena via Images

5.1 Introduction

Figure 5.1. The overall architecture of our overall learning-based solution to solving transparent fluid type phenomena. We present a network architecture capable of simultaneously predicting the dynamic 3D fluid surface and the distortion-free underwater pattern.

From our previous chapters, we have learnt that the 3D estimation from 2D images as well as distortion-free image recovery from highly distorted images due to refraction are inverse problems that are of great importance in the area of remote sensing, visualization, AR/VR, visual effects, photography, and computer games [63]. We also saw that the prior works for solving these inverse problems mainly required expensive manual setup and were incapable of working “in the wild” scenes. Based on our knowledge, there aren’t any work done to solve these two inverse problems simultaneously.
Figure 5.2. The overall network pipeline to solving dynamic and transparent fluid type phenomena.
In this chapter, we present a deep neural network which is capable of providing an overall solution to the transparent fluid type image related problems. Here, we simultaneously recover the transparent fluid surface as well as the underwater distortion-free image using a few temporal image sequence as input. We combine our neural network designs for 3D fluid surface reconstruction and the underwater distortion-free image recovery network by organizing the network layers and constraints such that the parameters get trained to predict both the unknowns simultaneously. Note that our Fluid Surface Reconstruction Network (FSRN) [111] requires known reference image as input which can now be provided by our distortion-free image recovery architecture. Also, note that the distortion map \( W \) can be computed from the heightmap \( H \) when the average surface height is known, 
\[
W = f(H) = \alpha \nabla H,
\]
where \( \nabla = \left[ \frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right] \) is the gradient operator, and \( \alpha = h_0(1 - \frac{1}{n}) \) is a constant scalar determined by the average surface height \( h_0 \) and the refractive index \( n \). Therefore, this dual relationship can be exploited to get the simultaneous 3D reconstruction and the distortion-free underwater pattern of a dynamic and transparent fluid media.

As shown in Fig. 5.2, our problem setup is a camera on the top looking at the dynamic fluid that has some texture pattern at the bottom. Our pipeline consists of two networks working hand-in-hand. We have the distortion-free image recovery network to recover the reference pattern taking a few temporal frames of refracted images as input, then we have a depth estimation network that takes the recovered reference pattern as well as the captured refracted image sequence as input to further estimate the 3D fluid surface geometry.

5.2 Real Results

In this section, we show some real results of our combined approach. Figure A shows multiple sequences of two underwater scenes as input. We compare our recovered distortion-free image with the true reference image and present the recovered depth-map, normal-map and the 3D surface of the input scene.
Figure 5.3. Real experimental results recovering the distortion-free image, depth-map, normal-map and surface 3D from the given input sequences.
Chapter 6.
Conclusions

In this chapter, we first conclude our research findings and then discuss the main research directions that we will be pursuing in our future work.

6.1 Summary

In this dissertation, we introduced four novel approaches to solve the dynamic, transparent fluid imaging problems. All the networks are physically constrained and agree with the law of light refraction in transparent media. The first network, dynamic fluid surface reconstruction network is a learning based deep neural network to recover the 3D fluid surface. We construct the supervised network with convolutional and recurrent layers to satisfy the temporal nature of the fluid motion. Similarly, the second network is also a supervised network for recovering the distortion-free underwater images. We also apply adversarial constraints to this network. We also create large-scale dataset to train our supervised networks. Our third network is more generalized non-rigid distortion removal network in both and air and water media. We apply unsupervised technique to train this network. Our first three networks extensively explore the temporal nature of the dynamic fluids. In our fourth approach, simultaneously perform recovery of the latent distortion-free image as well as 3D reconstruction of the transparent and dynamic fluid surface.

6.2 Future Directions

The following directions will extend our existing work to bridge several urgent gaps in the learning based dynamic fluid research. Simultaneous fluid surface and underwater scene reconstruction using deep neural network would be a good direction which could be obtained by combining our algorithms such that a single network can predict both the 3D fluid surface as well as the underwater scene. We can further, make the underwater scene non-planar, similar to real world to make the problem even more challenging. Further, we also extend our work to perform fluid surface and surrounding 3D scene reconstruction
from highly reflective fluid images, such as images taken of the natural lakes, ponds, etc. In terms of air turbulence, our work can be extended to detect motion from air-turbulent images using deep neural networks. For instance, moving vehicle in the turbulent traffic video, flying birds from the natural turbulent videos taken from long range cameras, etc. We summarize these directions as follows:

- Simultaneous fluid surface and underwater 3D scene (non-planar) reconstruction using deep neural network.
- Fluid surface and surrounding 3D scene reconstruction from highly reflective fluid images.
- Simultaneous Video Stabilization and Moving Object Detection in Turbulence using deep neural network.

These are some of the promising future directions of our work.
Appendix A. Copyright Information

This copyright is applied to Chapter 2 of this dissertation.
References

12th USENIX Symposium on Operating Systems Design and Implementation (OSDI


sequences with moving objects using recursive image fusion. In 2018 25th IEEE

absorption for depth recovery. In European Conference on Computer Vision, pages

joint material and lighting estimation. In Proceedings of the IEEE Conference on

normal prediction. In Conference on Computer Vision and Pattern Recognition, pages
5965–5974, 2016.


of directionally spread surface water waves. Journal of Computational Physics,


[10] A. Bulat, J. Yang, and G. Tzimiropoulos. To learn image super-resolution, use a gan
to learn how to do image degradation first. In Proceedings of the European conference
on computer vision (ECCV), pages 185–200, 2018.

movie for optical flow evaluation. In European conference on computer vision, pages

2014 22nd International Conference on Pattern Recognition, pages 883–888. IEEE,
2014.


Vita

Simron Thapa was born in Biratnagar sub-metropolitan city in Nepal. She received her Bachelor’s degree in Computer Engineering from Tribhuvan University in 2013 with academic scholarships in all her four years of college. She graduated as the top 1% student from the class of 2013. During her undergraduate days, she was recognized with multiple scholarships for academic excellence, and she represented the university in different events.

She initiated her professional career working as an Associate Software Engineer at Deerwalk Inc. She was soon promoted as a Software Engineer during her 18 months of industrial work where she developed applications for performing statistical analysis and visualization of US healthcare big datasets.

In the year 2015, she received a full assistantship from Louisiana State University (LSU) towards obtaining her graduate degree in Computer Science. During her Master’s degree, she worked in the Scientific Visualization lab with Dr. Bijaya B. Karki. During these years, she received a travel grant to attend SuperComputing Conference 2016. She was also active in various student organizations and held executive positions such as Graduate School Senator at Student Government, Membership Coordinator at Women in Computer Science, and Vice-president of Nepalese Student Association. She received her Master’s degree in Computer Science from Louisiana State University in August 2017.

Then, she continued her Ph.D. in Computer Science as a research assistant in Imaging and Vision Lab with Dr. Jinwei Ye. During these years, she became AnitaB.org’s Grace Hoppers 2020 scholar, 3M RISE 2020 scholar, and, ACM-W scholar to attend SIGGRAPH 2021. She also had two industrial summer internship experiences with Futurewei Technologies, Santa Clara, CA (2020) and Oppo US Research Center, Palo Alto, CA (2021). To date, she has her Ph.D. research work published in major computer science conference like IEEE/CVF Computer Vision and Pattern Recognition, 2020 as an Oral Presentation and two accepted in IEEE/CVF International Conference in Computer Vision, 2021. She expects to receive the Doctor of Philosophy degree in Computer Science in December 2021.