

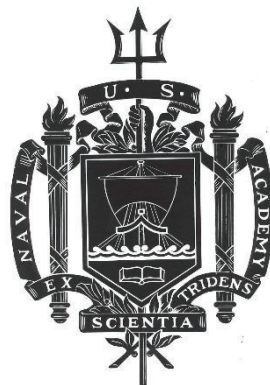
A TRIDENT SCHOLAR PROJECT REPORT

NO. 519

**Machine Learning-based Design of Structured Laser Light for Improved Data Transfer
Rate in Underwater Wireless Communication**

by

Midshipman 1/C William A. Jarrett, USN



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public
release and sale; its distribution is unlimited.

USNA-1531-2

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.				
1. REPORT DATE (DD-MM-YYYY) 5-16-22		2. REPORT TYPE		3. DATES COVERED (From - To)
4. TITLE AND SUBTITLE Machine Learning-based Design of Structured Laser Light for Improved Data Transfer Rate in Underwater Wireless Communication			5a. CONTRACT NUMBER	
			5b. GRANT NUMBER	
			5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) William A. Jarrett			5d. PROJECT NUMBER	
			5e. TASK NUMBER	
			5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Naval Academy Annapolis, MD 21402			10. SPONSOR/MONITOR'S ACRONYM(S)	
			11. SPONSOR/MONITOR'S REPORT NUMBER(S) Trident Scholar Report no. 519 (2022)	
12. DISTRIBUTION / AVAILABILITY STATEMENT This document has been approved for public release; its distribution is UNLIMITED.				
13. SUPPLEMENTARY NOTES				
14. ABSTRACT A system using Laguerre-Gaussian (LG) beams of structured light and deep convolutional neural networks (CNNs), is utilized. The structured beams of light are encoded to carry information, with each combination resulting in a distinct image. This creates an alphabet of 2^N images, where N is the number of basis beams and bits encoded per message. For this investigation, a novel methodology for optimizing network alphabet design is proposed, and 256 and 1024-beam alphabets are designed for optimal classification accuracy through optical turbulence using a CNN. For simulated evaluation, the beams were propagated using the split-step method through random phase screens drawn from the Nikishov spectrum for oceanic turbulence. In the experimental environment, the beams were propagated over ~2.5 meters through optically turbulent water with strong turbulent fluctuations. This study is novel in its use of the scintillation of a Gaussian beam to estimate the strength of the turbulence in real-time. In the simulated environment, we report 100% classification accuracy for the 256-beam alphabet, indicating the CNN's ability to learn weak fluctuations. Under experimental conditions, we report over 97% accuracy for 256-beam alphabets, and over 90% accuracy for the 1024-beam alphabet.				
15. SUBJECT TERMS Laguerre-Gaussian beams, orbital angular momentum, convolutional neural network, underwater wireless optical communication, optical turbulence, split-step wave simulation				
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 115
a. REPORT	b. ABSTRACT	c. THIS PAGE		
				19a. NAME OF RESPONSIBLE PERSON
				19b. TELEPHONE NUMBER (include area code)

U.S.N.A. --- Trident Scholar project report; no. 519 (2022)

**MACHINE LEARNING-BASED DESIGN OF STRUCTURED LASER LIGHT FOR
IMPROVED DATA TRANSFER RATE IN UNDERWATER WIRELESS
COMMUNICATION**

by

Midshipman 1/C William A. Jarrett
United States Naval Academy
Annapolis, Maryland

(signature)

Certification of Adviser(s) Approval

Professor Svetlana Avramov-Zamurovic
Weapons, Robotics, and Control Engineering Department

(signature)

(date)

Professor Joel M. Esposito
Weapons, Robotics, and Control Engineering Department

(signature)

(date)

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

(signature)

(date)

ABSTRACT

To meet the demand for high data transfer rate underwater wireless communication systems, a system using Laguerre-Gaussian (LG) beams of structured light and deep convolutional neural networks (CNNs), is proposed [1]. In this system, the structured beams of light are encoded through superposition to carry information, possible due to the orthogonality of the LG beam, with each combination resulting in a distinct image. This creates an alphabet of 2^N images, where N is the number of basis beams and bits encoded per message. Previous work has evaluated network performance using alphabets ranging from 16 beams to 32 beams. For this investigation, a novel methodology for optimizing network alphabet design is proposed, and 256 and 1024-beam alphabets are designed for optimal classification accuracy through optical turbulence using a CNN. The alphabets were evaluated in the simulated and experimental environments. For the simulated environment, the beams were propagated using the split-step method through random phase screens drawn from the Nikishov spectrum for oceanic turbulence. In the experimental environment, the beams were propagated over ~ 2.5 meters through optically turbulent water with strong turbulent fluctuations. This study is novel in its use of the scintillation of a Gaussian beam to estimate the strength of the turbulence in real-time. In the simulated environment, we report 100% classification accuracy for the 256-beam alphabet, indicating the CNN's ability to learn weak fluctuations. Under experimental conditions, we report over 97% accuracy for 256-beam alphabets, and over 90% accuracy for the 1024-beam alphabet. We find that our CNN is capable of classifying the alphabet symbols provided sufficient training data is available and representative of the testing data. A greater challenge arises when the network is tested on images with altered beam alignment.

KEYWORDS

Laguerre-Gaussian beams, orbital angular momentum, convolutional neural network, underwater wireless optical communication, optical turbulence, split-step wave simulation

ACKNOWLEDGEMENTS

To Dr. Svetlana Avramov-Zamurovic, my primary advisor, who has been a great guide and mentor, in this project, but also in all things related to life. I am incredibly grateful for all the time you have given me, and could not have been nearly as successful without the hours of academic and emotional support you gave me. Thank you!

To Dr. Joel Esposito, who welcomed me to this project right away. Your willingness to teach me and help me learn about all things neural networks has been incredible, and I am very grateful for the hours of time you have given me.

To Dr. Peter Judd, Dr. Ryan Lindle, and Dr. Abbie Watnik at the Naval Research Laboratory, Washington, D.C. for inviting me back this semester to help build the Rayleigh-Bénard turbulence tank, and for their general project support, guidance, and mentorship.

To Mr. Dan Rhodes and Mr. Joe Bradshaw from the Technical Services Division for their invaluable help in setting up the underwater turbulence tank and device synchronization.

To Dr. Charles Nelson for his advice, mentorship, and patience when helping me with optical equipment in the lab and project ideas.

TABLE OF CONTENTS

Project Abstract and Keywords	1
Acknowledgments	2
Trident Scholar Report	
1. Project Introduction	5
2. Motivation for Research.....	7
3. Background Information for UWOC System	7
A. Transmission - Laguerre-Gaussian Beams, Orbital Angular Momentum of Light, and Beam Generation	8
B. Reception and Classification - Deep Convolutional Neural Networks	9
C. Propagation - Characterization of the Experimental Turbulent Underwater Medium	10
4. Data Collection and Methods of Evaluation.....	13
A. Experimental Set-Up and Data Collection	13
B. Data Processing	14
C. Optical Split-Step Simulation	15
5. State of the Art in Published Literature	26
A. Resilience of Beams Carrying OAM Propagating through Complex Environments	26
B. Integrating CNNs into UWOC Systems Utilizing Beams Carrying OAM	26
C. Statement of Novelty	28
6. Comparison of Structured Light for SPIE Remote Sensing 2021	29
A. Generation of Structured Light	29
B. Experimental Set-Up	30
C. Use of Deep Convolutional Neural Network	33
D. Results and Conclusion	33
7. Basis Beam Selection	34
A. Evaluation through Simulation.....	35
B. Evaluation through Hybrid Set-Up.....	36
C. Experimental Evaluation of Basis Beams	40
D. Novel Basis Beam Selection Methodology	44
8. Improving Data Transfer Rate	45
A. 64-beam Alphabet.....	45
B. Design of 256-beam Alphabets	45
C. Evaluation of 256-beam Alphabets through Simulation	49
D. Experimental Evaluation of 256-beam Alphabets and Turbulence Validation	49
E. 256-beam Alphabet Results and Conclusions	53
F. 1024-beam Alphabet Evaluation and Performance	55
9. Beam Misalignment	60
A. Design and Set-Up of Experiment	60
B. CNN Performance	61
C. Conclusions and Next Steps	62

10. Construction of a Rayleigh-Bénard Turbulence Tank at Naval Research Laboratory, Washington, D.C.	63
A. Rayleigh-Bénard Turbulence	63
B. Tank Construction	64
C. Future Experiments	66
11. Conclusions and Future Work	66
A. Conclusions	66
B. Lessons Learned	67
C. Future Work and Ongoing Research	67
12. References	68
13. Appendix	72
A. MATLAB Code Written for Trident Scholar Project.....	72
1. LGbeamsforML.m	72
2. LGBeamCombinations.m	77
3. Beamset_Corr.m	80
4. genLG.m	81
5. Tiff_Processor.m	82
6. CNN.m	83
7. Calc_SI.m	85
8. Nikishov_Sim.m	89
9. genSLMTurbScrns.m	94
10. ft_nikishovPS.m	95
11. sh_nikishovPS.m	97
12. ft_sh_nikishovPS.m	98
13. Screen_Vali.m	99
B. 2021 SPIE Remote Sensing Publication	102
C. 2022 Conference on Lasers and Electro-Optics 2-Page Summary	111
D. 2022 Optica Imaging and Applied Optics Conference 2-Page Summary	113
E. 2022 SPIE Optics and Photonics Abstract	115

Section 1: Project Introduction

Communication systems are an integral part of our daily lives. Every day, an incredible amount of information is transmitted to and from our phones, computers, and other devices. Because of this, communication systems are integral in numerous areas, including the military. Communication can be broken into three main facets: the generation of a signal, the propagation of the signal through a medium, and the reception and classification of the signal. This can be understood via the simple example of a conversation. When we converse, one person creates a signal in the form of sound waves generated by the vocal chords. This sound wave propagates through the air to the other person's ear, where it is received and then classified by the brain as words, establishing a communication link. If at any point along the process something goes wrong, a link cannot be established. Examples of this include: a person speaking too quietly (issue at the transmitter), a noisy room (environmental signal degradation), hearing loss (issue at the receiver), and speaking different languages (signal classification issue). In all of these cases, communication is difficult or impossible.

In this Trident Scholar project, a novel underwater wireless optical communication (UWOC) system [1] is investigated and further developed for practical use. The basic architecture of the system is shown in Fig. 1.1, which broadly illustrates how the system transmits information. The details for each step along the system's communication path will be discussed in this report.

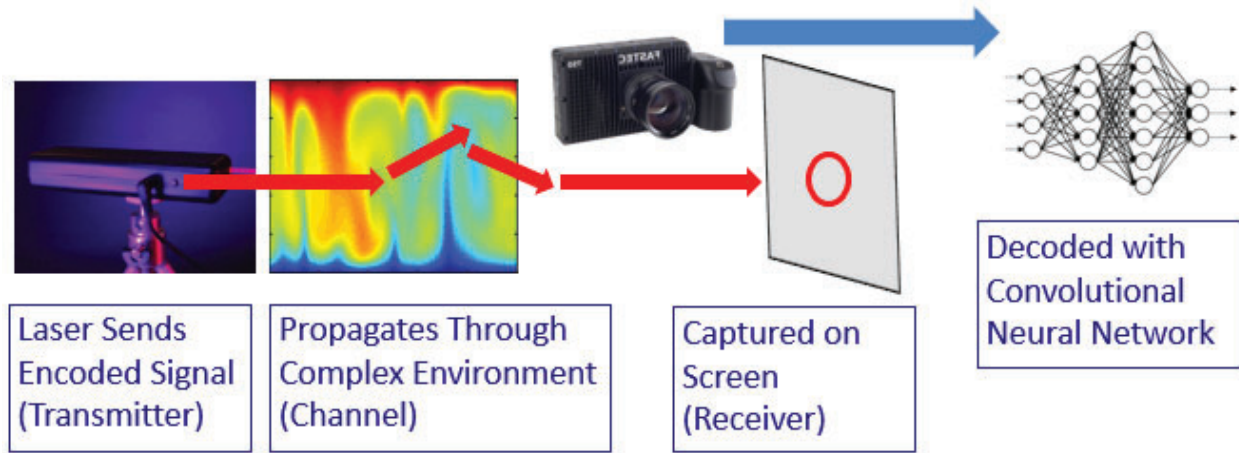


Figure 1.1: Proposed communication system overview

This research seeks to help meet the demand for high-speed underwater wireless communication systems by making use of structured beams of light that are spatially encoded to carry more information, maintaining a resistance to underwater attenuation and optical turbulence, and providing secure transmission. The spatial encoding is the result of combining basis beams, with each combination resulting in a distinct intensity pattern (Fig. 1.2A). The resulting alphabet of images, such as in Fig. 1.2B, can then be propagated through the underwater medium, and decoded

by a machine learning (ML) neural network. Fig. 1.2C shows the experimentally captured images of the beams after propagation through an optically turbulent underwater environment.

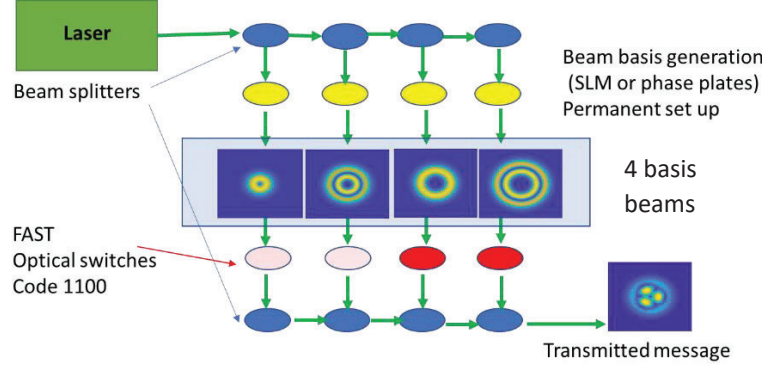
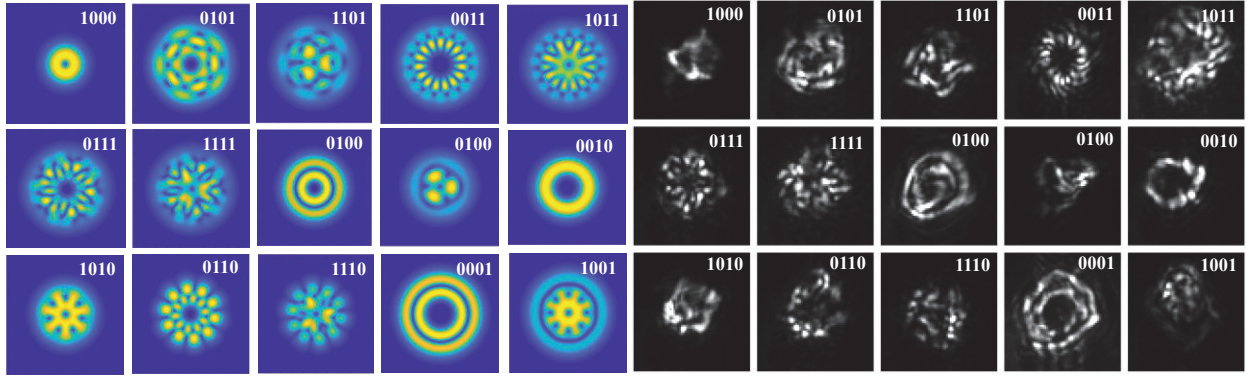


Figure 1.2A: Demonstration of basis beam superposition to create unique messages [2]

(a)



(b)

(c)

Figure 1.2B: Theoretical beams created using four basis beams [2]

Figure 1.2C: Experimental beams captured after propagation through underwater medium with optical turbulence present [2]

In order to extend the operational range of the proposed system, the structure of the basis beams will be optimized for the ML network, in the context of optical turbulence generated as the result of fluctuations in water temperature. This research seeks to increase the data transfer rate of the system by creating a reliably classifiable alphabet of beams that are resilient to the underwater environment, while increasing the system data rate by progressively expanding the number of basis beams to ten, versus four (Fig. 1.2A) [2]. This project benefits the field of UWOC through an investigation into the mitigation of challenges faced in the underwater environment, and through further development of a high bandwidth UWOC system. It also seeks to quantify the performance of the system using a Bit Error Rating (BER). This metric describes the number of errors per messages sent in a communication system, regardless of the reason for the said error. In the industry today, the standard is usually considered to be a BER of 1 error per 100,000 messages sent, or that the system works 99.999% of the time [3].

Section 2: Motivation for Research

There is an increasing need for high bandwidth underwater wireless data transmission for military operations, tactical surveillance, pollution monitoring, oil rig control, offshore explorations, environmental monitoring, and oceanography [4] [5] [6] [7] [8]. There is not as strong of a demand for wired communication because it comes with the inescapable drawback that a communication link requires a physical connection. This is impractical for many applications necessitating mobile high-speed communication such as communication with a submarine or autonomous underwater vehicle (AUV). It is also impractical for episodic connections, such as transmitting data from a sensor node to an AUV, given the inherent difficulty physically connecting in the underwater environment. As such, most research focuses on improving wireless communication given its much greater versatility [4]. This has caused researchers to turn to underwater wireless optical communication (UWOC) systems, because other commonly used methods, such as acoustic and radio-frequency waves, are limited in bandwidth and are often unsecure due to omnidirectional propagation [6] [9] [10]. This is summarized in Fig 2.1.

Parameters	Acoustic	Radio Frequency	Optical
Bandwidth	Distance dependent: < 100 m $\approx$ 100 kHz (0.1 MHz)	$\approx$ 1 MHz	$\approx$ 100 MHz

Figure 2.1: Comparison of different types of underwater wireless communication

While UWOC provides a solution to many of the problems posed with acoustic and RF communication, it does face its own challenges, primarily attenuation and optical turbulence. Due to a much greater frequency, the main sources of attenuation of optical waves in seawater are not propagation in a uniform conductor [4], but rather absorption and scattering. These are the result of the molecules and large particles, such as salts, minerals, or organic particles, that are present in the medium that absorb and scatter the photons of light, limiting propagation distance. In UWOC, light in the blue-green part of the spectrum is typically used because this range of wavelengths (from 450-500 nm) has been shown to have the least attenuation in the underwater environment [6]. Optical turbulence is caused by changes to the temperature, salinity, and density of the medium that result in small changes to the refractive index of the medium [6]. These changes in the refractive index cause the light to bend away from the intended target, as described by Snell's Law, and result in laser light phase fluctuations and intensity variations. Due to these effects, practical UWOC is currently limited to distances up to 100 meters [6].

Section 3: Background Information for UWOC System

In the following subsections, this report will identify and discuss relevant details regarding our communication system's transmission of the information (Sect. 3A), reception and classification of the signal (Sect. 3B), and characterization of the transmission channel (Sect. 3C).

Section 3A: Transmission - Laguerre-Gaussian Beams, Orbital Angular Momentum of Light, and Beam Generation

The signal used in this research is a laser beam that will be modeled as a higher-order mode Gaussian beam, known as a Laguerre-Gaussian (LG) beam. LG beams are of interest because they carry internal orbital angular momentum (OAM). The OAM of light is the component of angular momentum of light that is dependent on the field spatial distribution, or where the light is in space. In LG beams, this comes from the helical waveform of the light, giving the light a corkscrew-like shape as it propagates. This produces a “donut-like” intensity pattern (Fig. 3.1A) with a phase change that is shown in Fig. 3.1B. The helical nature of the beam can also be seen in the equation that governs it (Eq. 3.1), in the form $e^{im\theta}$. The expression for the complex amplitude, $U_{n,m}(\rho, \theta, z)$ given in cylindrical coordinates in Eq. 3.1 is used to calculate the magnitude and phase for a LG optical beam at any distance along the axis of propagation, z , where ρ is the radius and θ describes the angle of the waveform [11] [12]. In Eq. 3.1, I_0 is the initial intensity (usually normalized to 1), ω_0 is the beam waist, $\omega(z)$ is the beam radius, k is the optical wave number, z_R is the Rayleigh range, n is the order of the Laguerre polynomial, and m is the topological charge [13].

$$U_{n,m}(\rho, \theta, z) = \sqrt{I_0} \frac{\omega_0}{\omega(z)} e^{-\frac{\rho^2}{\omega(z)^2}} e^{-i\frac{k\rho^2 z}{2(z^2 + z_R^2)}} e^{-ikz} \sqrt{\frac{2n!}{\pi\omega_0^2(n+|m|)!}} \\ \times \left(\frac{\sqrt{2}\rho}{\omega(z)}\right)^{|m|} L_n^{|m|}\left(\frac{2\rho^2}{\omega(z)^2}\right) e^{-i(2n+|m|+1)\arctan\left(\frac{z}{z_R} + im\theta\right)} \quad (3.1)$$

The topological charge, m , characterizes the helical waveform and corresponding optical vortex of the beam. It describes the number of times that the light rotates about the axis in one wavelength; therefore, the higher the topological charge, the faster the light is rotating, and the greater OAM it carries. When presented as a cross-section of the intensity, the orbital vortex will appear as a dark “hole” in the center of the light, due to the light at the axis interfering in a destructive manner. As the topological charge increases, the radius of the orbital vortex increases. The order of the Laguerre polynomial, n , describes the number of secondary rings in the resulting intensity pattern. This is significant for communication because it allows for the creation of an “alphabet” of unique images, meaning many bits of information can be passed with a single symbol. The different intensity patterns can be associated with data, propagated through a complex medium, captured by a camera, and processed by a computer [13], a system which forms the basis of the one used in this research. The alphabet can be created using a varying number of basis beams, as shown by the use of multiple beam sets and differing numbers of base beams in [2]. The number of unique image intensity patterns $N_{images} = 2^n$, where n is the number of base beams.

In order to create the LG beams, a Spatial Light Modulator (SLM) was used. Our primary SLM is an electrically addressed liquid-crystal Meadowlark Optics 1920 x 1152-pixel phase-only SLM, meaning it modulates the phase of the incoming light by changing the refractive index of the liquid-crystal layer. Each pixel is controlled by an independent electrode, which is set to the desired voltage as provided by the controlling computer. SLMs are extremely useful for this proof-

of-concept research as they allow us to quickly and easily generate a large variety of basis and superimposed LG beams to test our alphabets. Fig. 3.1A represents a sample SLM screen used to create a LG beam with a topological charge of 1, and an order of 0, shown in Fig. 3.1B. The pixelated grating allows us to selection for the desired spatial mode, typically the first order mode, and the phase fork is what “spins” the beam and provides it with OAM. The phase resulting phase of the light is provided in Fig. 3.1C, where one can see the rotation. Since this beam has a topological charge of one, it makes one complete rotation. The SLM screens are prepared in MATLAB using “LGbeamsforML.m”, located in Appendix A, Sect. 1.

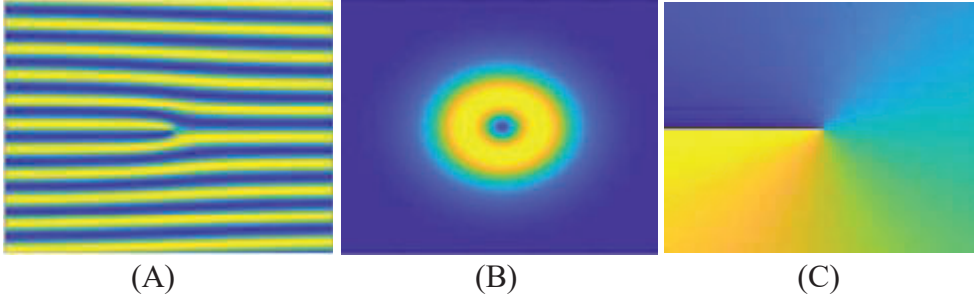


Figure 3.1: (A) SLM Screen for LG Beam $U_{0,1}$ from Eq. 3.1, (B) resulting intensity pattern of beam $U_{0,1}$ when Gaussian beam is reflected off SLM with screen (A), and (C) resulting phase of beam $U_{0,1}$

Section 3B: Reception and Classification – Deep Convolutional Neural Networks

The main purpose of using light carrying OAM is that information can be encoded, or multiplexed, into the structure of the light. Multiplexing is a communication strategy that increases the communication data rate by combining multiple signals into one so that information can be passed more effectively. There are many ways of multiplexing and demultiplexing information when using beams carrying OAM, but in this research, only image classification will be explored. LG beams can be used to create images that contain multiplexed information, which then must be accurately and reliably classified even in the presence of significant attenuation and optical turbulence. The larger the beam alphabet, the greater the bandwidth of the system, but the more difficult the classification problem becomes. In recent times, deep convolutional neural networks (CNNs) have proven to be extremely effective in image classification.

CNNs are a type of neural network specifically designed for image classification, as the traditional machine learning model of numbers as independent inputs and outputs cannot be used because the spatial structure of the pixels contains necessary information for understanding the image as a whole. In a CNN, each layer takes in a set of images as the input (green), and produces a set of images as an output, preserving the proximity relationship of the pixels. CNNs are typically composed of many sets of three layers: a 2-D convolutional layer (orange), a rectified linear unit layer (yellow), and a max pooling layer (gray). The outputs from these layers are then used to classify the image in the output layers (red). Further information regarding each layer can be found in Ref. [14] [15] [16]. For this research, a CNN with the same architecture (shown in Fig. 3.2) as the one used in Ref. [2] and [17] will be used.

Layer	Type	Details
1	Input	64 x 64 grayscale image
2	2D Convolution	8 3 x 3 filters, stride 1, padding “same”
3	Batch Normalization	
4	Rectified Linear Units	
5	Max Pooling	2 x 2 max pooling, stride 2
6	2D Convolution	8 channels, 16 3 x 3 filters, stride 1, padding “same”
7	Batch Normalization	
8	Rectified Linear Units	
9	Max Pooling	2 x 2 max pooling, stride 2
10	2D Convolution	16 channels, 32 3 x 3 filters, stride 1, padding “same”
11	Batch Normalization	
12	Rectified Linear Units	
13	Fully Connected	3 outputs
14	Soft Max	
15	Classification	3 classes

Figure 3.2: CNN architecture [17]

For our work, the CNN was designed, trained, and used to test the images using MATLAB 2021b and the Deep Learning Toolbox. The images were randomly assigned to the training, validation, and testing sets, with typically 40% being used for training, 50% being used for testing, and 10% used for validation, though this varies by experiment. Training used the stochastic gradient descent method with momentum of 0.5, an initial learn rate of 0.01, and mini batch size of 64. The training was typically terminated after a maximum of 20 failed validation checks.

Deep CNNs are extremely effective at image classification problems as they bypass the need for human intervention in feature determination. Deep CNNs provide an effective and reliable method for classifying images of beams carrying OAM, even in high attenuation and optical turbulence [1] [2] [17] [18] [19] [20]. All CNN work was completed using the “CNN.m” script provided in Appendix A, Sect. 6.

Section 3C: Propagation – Characterization of the Experimental Turbulent Underwater Medium

While the propagation of a beam of light through a vacuum can be described using the paraxial wave equation, in a realistic environment such as the ocean, attenuation, due to absorption and scattering, and optical turbulence will affect the propagation of the light. The varying salinity, pressure, temperature, and turbidity (motion of organic and inorganic particles) of ocean water will distort the light beam and make establishing a successful link more challenging [21]. Characterizing and modeling the underwater channel is fundamental to improving UWOC. The

strength of the distortions due to optical turbulence can be related to the refractive-index structure constant, C_n^2 . There are several methods of estimating this number, primarily through simulation, experimental measurements, or scintillation of a Gaussian beam.

In previous work through our tank, the turbulence was estimated using the Multiphysics program COMSOL and temperature recordings from thermocouples placed outside the propagation path of the beam [2]. COMSOL simulated the heater as a point source, and calculated temperature changes due to natural convection inside the finite area, using this to estimate the strength of the fluctuations of the refractive index. For the experimental work, the temperature variations and differences at the thermocouples was used to estimate the strength of the refractive index fluctuations. From both sources, the value for the refractive index structure constant (C_n^2) ranged from $1 \times 10^{-7} - 1 \times 10^{-8} \left[m^{-\frac{2}{3}} \right]$, which indicates fluctuations in the strong turbulence regime. The drawback to these methods of C_n^2 estimation is that they don't accurately capture the variation and extent of the turbulent fluctuations. COMSOL simplifies the geometry and physics of the heater and environment significantly in order to be able to calculate the convection, while the thermocouples are placed every inch, and in the beam's path, forming an obtrusive and crude grid. This led us to explore the use of scintillation of a Gaussian beam to more accurately estimate the turbulence experienced by our beams. For this method, the scintillation index (SI) of a Gaussian beam was calculated from normalized images of the Gaussian beam spanning the total collection time. This was then used to find the approximate refractive-index structure constant C_n^2 per experiment, following the methodology given in Ref. [22] and [23], and based on the derivation given in Ref. [24]. Eq. 3.2 was used to determine the SI of the beam at each pixel of the Gaussian beam, spanning the total collection time, where σ_B^2 is the SI and I represents the normalized intensity value of the pixels in the Gaussian beam.

$$\sigma_B^2 = \frac{\langle I^2 \rangle - \langle I \rangle^2}{\langle I \rangle^2} \quad (3.2)$$

Eq. 3.3 was then used to find the Rytov variance, σ_R^2 , where Θ is the output plane refraction parameter and Λ is the output plane diffraction parameter [23] [24].

$$\sigma_R^2 \cong \frac{\sigma_B^2}{3.86 \left\{ 0.40[(1 + 2\Theta)^2 + 4\Lambda^2]^{5/12} \cos \left[\frac{5}{6} \tan^{-1} \left(\frac{1 + 2\Theta}{2\Lambda} \right) \right] - \frac{11}{16} \Lambda^{5/6} \right\}} \quad (3.3)$$

From Eq. 3.3, the refractive-index structure constant, C_n^2 was estimated in Eq. 3.4, where k is the wave number and z is the total propagation distance.

$$C_n^2 \cong \frac{\sigma_R^2}{1.23 k^{7/6} z^{11/6}} \quad (3.4)$$

For every experimental case where this method was used, the normalized mean intensity of the Gaussian beam was plotted to ensure valid data collection and to find the centroid for spatial averaging. An example of this is given below in Fig. 3.3A, where the time-averaged beam from a single experiment makes a clear Gaussian shape. This tells us that even in the presence of

turbulence, the beam is still averaging the form we would expect. Fig 3.3B provides an example of the same Gaussian beam from one instant, where there is no induced optical turbulence impacting the beam. Then Fig. 3.3C provides a counter example of the same beam from the same experiment, but in a moment of strong induced turbulence, in order to illustrate the experimental variation.

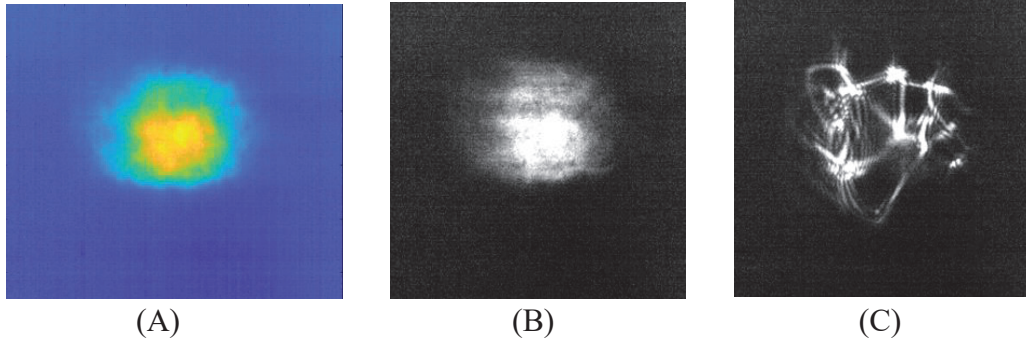


Figure 3.3: (A) Time-averaged and normalized mean intensity of a Gaussian beam, (B) Gaussian beam under no induced optical turbulence, and (C) Gaussian beam under strong induced optical turbulence

From the time-averaged intensity value $\langle I \rangle$ of the Gaussian beam, we can also use Eq. 3.2 to calculate the pixel-by-pixel SI. This is useful because it provides information regarding where on the beam the turbulence is strongest, as greater scintillation corresponds with greater turbulence. This is shown below in Fig 3.4, for the same experiment and beam shown in Fig. 3.3. As can be seen, the scintillation is varied, but relatively contained within the radius of the beam. Additionally, the strongest scintillation occurs at the bottom of the beam, which validates the heater used, which is placed approximately 5 cm below the beam and produces turbulent fluctuations that flow upward in the y-direction due to natural convection. We also see a reduction in scintillation in the center of the beam, which is expected from previous literature [22]. This center area is what is used to estimate C_n^2 , from a spatial averaging of the center 21 x 21 pixels.

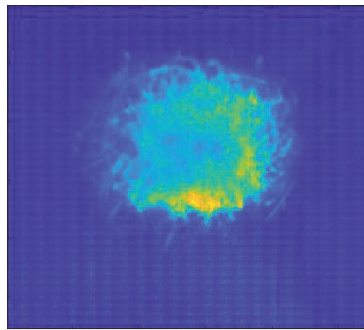


Figure 3.4: Time-averaged SI of a Gaussian beam

Proper characterization of the turbulent medium is essential to understanding how our messages are impacted, and under what conditions this system could practically perform. Turbulence estimation is important because it allows us to validate that our experiments are maintaining similar levels of optical turbulence, reducing the unknown variables of our experiments. Additionally, experimenting under different levels of turbulence will offer the data to determine the effectiveness of the system in various conditions, while informing how to design the beams

and best utilize the machine learning network. This led us to develop several methods for evaluating our beams, in order to provide a wide variety of environments and turbulence strength. The processing and estimation of the refractive index structure constant from the SI was completed in MATLAB using “Calc_Sl.m” located in Appendix A, Sect. 7.

Section 4: Data Collection and Methods of Evaluation

Section 4A: Experimental Set-Up and Data Collection

Our setup is given in Fig. 4.1 with power supply (A), stabilized 2 mW, 632.8 nm He-Ne laser (B), beam expander (C), spatial light modulator (SLM) (D), mobile workstations ($E_{1,2}$), water tank (F), thermal generator (G), lens (H), display screen (I), and camera (J). The LG beams are generated using a 1920 x 1152-pixel SLM. They are then propagated ~ 2.5 meters through the tank (F), where they are projected onto the screen (I), and captured by the TS5 Fastec high-speed camera (J). For each physical experiment run, the camera was typically set to 275 fps, a resolution of 320 x 320, and a bit-depth of 8. This allowed for the images to be captured at a high resolution while decreasing dataset size for easier download and post-processing.

Following the experiments detailed in Sect. 6, we quickly realized that it would be necessary to synchronize and cycle the camera and SLM when generating large alphabets. In the beginning of this project, we would manually cycle through the beams one at a time, collecting 1000 images of each beam while the heater in the tank continuously produced optical turbulence. This created a delay as the images were downloaded from the camera to our hard drive, making the process slow and laborious, as each beam took approximately 2-3 minutes. This may sound small, but for 256 or 1024 beam alphabets this would require active data collection of 10-24 hours. Most importantly though, in previous experiments [2] [25], it has been shown the turbulent flow from our heater changes rapidly and unpredictably, flowing on the scale size of approximately 1 cm [14], meaning one beam might experience significantly different turbulence from another resulting in less accurately representative data. These challenges necessitated cycling the beams on the SLM, so that data collection could be done in one collection per experiment, and all the beams experienced the full scope of turbulence. Synchronization of the SLM and camera allowed us to collect a 1024-beam dataset, as well as mitigating the impact of temperature changes in the tank over time. This minimized experimental inconsistencies.

In order to cycle the beams on the SLM, we also needed to synchronize the camera and SLM. This ensured that there was no crossover (synchronization error) as the SLM switched between beams. To accomplish this, a trigger signal was generated using a function generator and sent to both the SLM and camera simultaneously using BNC cables, as shown in Fig. 4.2.

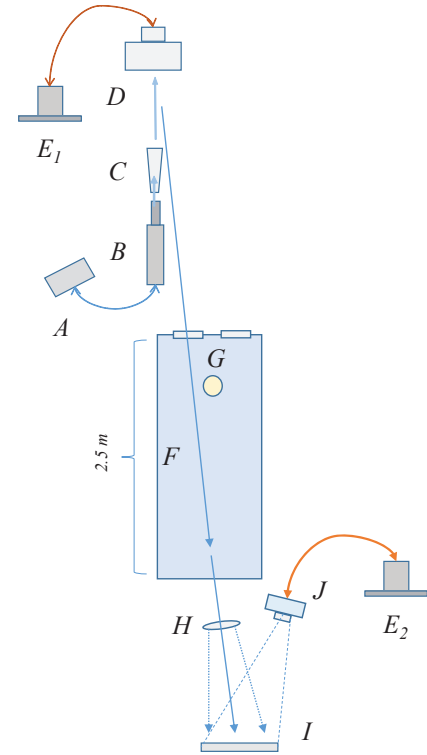


Figure 4.1: Experimental Setup [25]

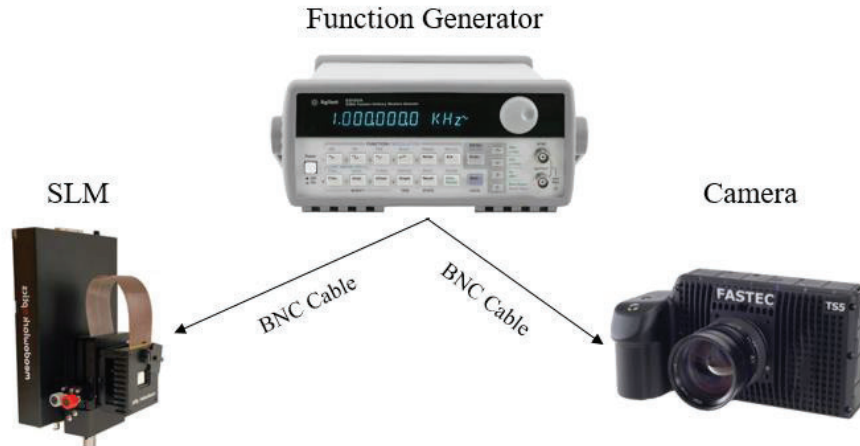


Figure 4.2: Synchronization of SLM and Camera

It was a simple task to set the SLM up for synchronization, as there are built in settings to cycle the SLM upon receiving an external trigger. This can be accomplished in the main program that runs the SLM, BlinkOverDrive plus [26]. However, in order to produce desired results from the camera, it was necessary to manipulate the settings in a non-intuitive way. When recording, the camera has two modes: armed and triggered. The camera can be set up to receive an external trigger, given by the function generator, but traditional usage of the camera only allows for one trigger per recording session. This is not practical for our usage of system, as we need to collect 5-10 images per trigger for a long period of time. To get around this, the camera needed to be put into Long Record (LR) mode, to make use of the Burst Record on Command feature. This feature allows a set number of frames, dictated prior to starting the record session, to be taken each time the camera is triggered [27]. Though this seems straight-forward, engineering a solution to the camera and SLM synchronization problem was a significant challenge. However, with this system setup, the data collected through turbulence provided better results because each beam in the alphabet had exposure to the full scope of the turbulence, with samples from the entire timescale, and decreased the long-term experimental labor significantly.

In each experiment, the beams were cycled on the SLM at a rate of 20 Hz, meaning the beam pattern was changed every 50 ms. Within each cycle, 11 images of the current beam were collected by the camera, totaling 40 ms for each collection, as the camera collected at 275 fps. The extra 10 ms were added to the end in order to minimize capture error by the camera. However, when the data was processed, the 11th frame of each collection was cut out to minimize synchronization error and crossover between adjacent beams as the SLM switched. For each experiment, we made 100 full loops of the experimental beam set, collecting 1100 images of each beam, or 1000 usable images after post-processing. This resulted in the large image sets, which had to be processed and managed, which will be the focus of the next section.

Section 4B: Data Processing

The raw images collected experimentally were saved as full TIFF stacks, in order to ensure all variations in the pixel intensity data was saved. These images were then imported and processed in MATLAB using “Tiff_Processor.m” provided in Appendix A, Sect. 5. The general

processing applied to each image follows the below outline given in Table 4.1. Sample images of the raw and processed are provided in Fig. 4.3, demonstrating the difference. Typical processing time was 16,000-18,000 images per hour, but in a practical system, this processing could be built into the receiver, and would not create a data transfer rate choke point.

Step	Description
imread	Reads in image file in uint8 format (8 bits)
im2gray	Converts image to grayscale [320 x 320 x 1]
imadjust	Adjusts contrast of image to predetermined amount, dependent on experiment, to ensure all background noise was dropped and the beam intensity was readable for the CNN. It was important to select my own thresholding in order to minimize background noise, rather than use MATLAB's auto adjust
imcrop	Crops image if necessary to remove undesired artefacts or features in experiment
imresize	Resizes image to 64 x 64 pixels using bicubic interpolation for faster processing, reupload, and CNN training / testing
imwrite	Resaves image to hard drive

Table 4.1: Description of MATLAB functions used to process images for CNN evaluation

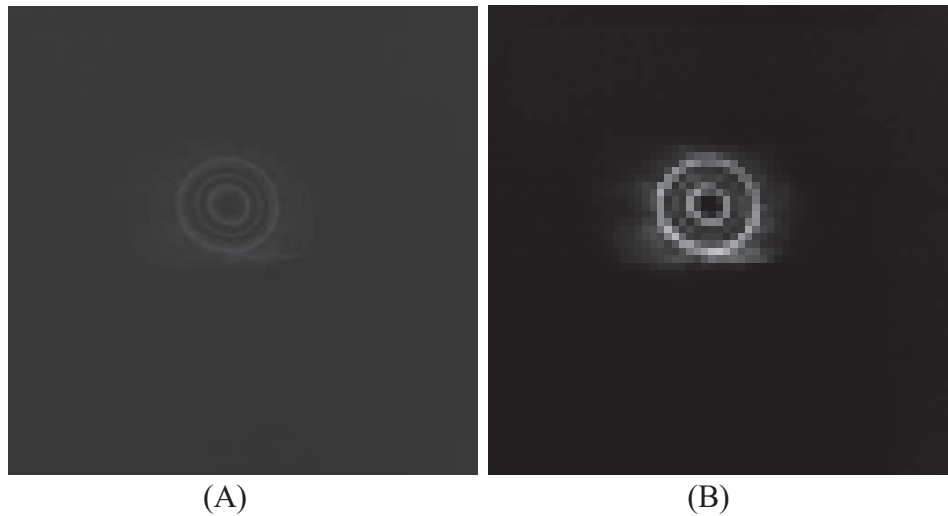


Figure 4.3: (A) Raw image of LG beam $U_{1,4}$, 320 x 320 pixels and (B) processed image of the same at 64 x 64 pixels

Section 4C: Optical Split-Step Simulation

In order to add variety to the conditions we could evaluate our beams through, I decided to add a simulated component to my project and analysis. Modeling a turbulent environment is challenging due to the unpredictable changes that occur in both time and space. Therefore, it is far easier to model turbulence by treating it stochastically, as a random process that is both homogenous and isotropic, following Kolmogorov turbulence theory [24]. This problem can then be further simplified by taking a continuous environment, and representing it in the form of discrete phase screens, which distort an optical wave only at that point in time and space. Between the

phase screens, the optical wave propagates in a vacuum. This is the basis for the split-step optical wave simulation, which models the complex medium in the form of thin phase screens placed along the path of propagation, shown in Fig. 4.4. The script to run the simulation, “Nikishov_Sim.m” is provided in Appendix A, Sect. 8.

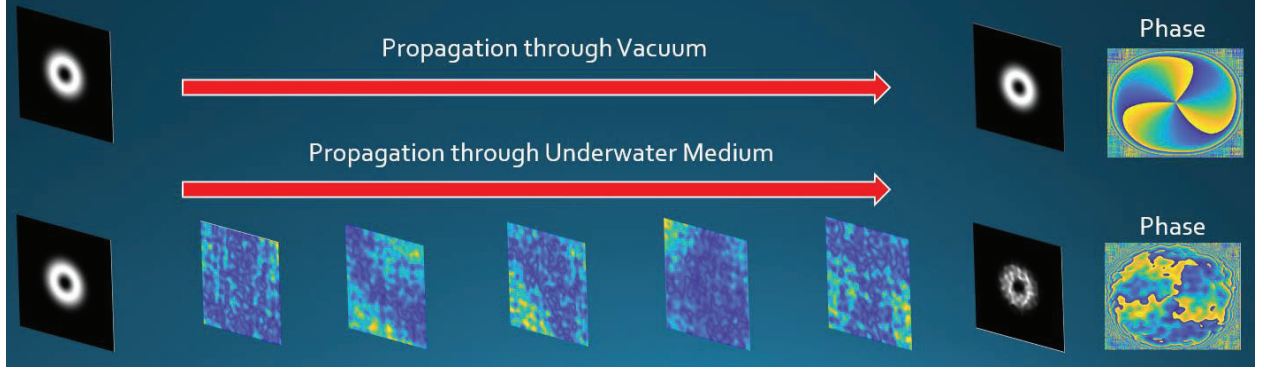


Figure 4.4: Comparison of Optical Wave Simulation for Vacuum and Complex Medium

Though the split-step method is useful for simulating accurate optical wave propagation with relative ease, it is not without its drawbacks. Primarily, the use of Kolmogorov theory of turbulence limits the strength of turbulence of each screen to weak fluctuations of the propagating beam [24]. In order to quantify the strength of the turbulence using the scintillation in the propagating beam, we make use of the Rytov method, assuming a line-of-sight system. For the Rytov method, the optical field can be written as shown in Eq. 4.1, where $U_0(\mathbf{r})$ represents the propagation of the beam in a vacuum, and $\psi(\mathbf{r})$ is the complex phase perturbation. The complex phase perturbation can be written as shown in Eq. 4.2 to calculate successive perturbations, such as the perturbation that is the result of each discrete screen. The complex phase perturbation can also be written in terms of the log-amplitude perturbation χ , and phase perturbation ϕ , given in Eq. 4.3 [28].

$$U(\mathbf{r}) = U_0(\mathbf{r})e^{\psi(\mathbf{r})} \quad (4.1)$$

$$\psi(\mathbf{r}) = \psi_1(\mathbf{r}) + \psi_2(\mathbf{r}) + \dots \quad (4.2)$$

$$\psi(\mathbf{r}) = \chi + i\phi \quad (4.3)$$

The Rytov method allows us to compute the total log-amplitude variance $\sigma_{\chi,sw}^2$, or Rytov variance, of a propagating spherical wave using Eq. 4.4. The Rytov variance is a function of the wave number k , the total propagation distance Z , the propagation distance z , the number of screens n , and the refractive-index structure constant C_n^2 . The turbulence level must result in a total Rytov variance of less than 0.25, with each screen having a Rytov variance of less than 0.1, providing the first simulation constraint. The strength of turbulence in simulation is controlled by the rate of dissipation of kinetic energy per unit mass of fluid ε , the rate of dissipation of mean-squared

temperature χ_T , which are used to estimate C_n^2 via Eq. 5.5, where the constant $\alpha_T = 2.4 \times 10^{-4} \left[\frac{m}{K} \right]$ [29].

$$\sigma_{\chi,sw}^2 = 0.563 k^{7/6} \Delta z^{5/6} \sum_{i=1}^n C_{ni}^2 \left(\frac{z_i}{Z} \right)^{5/6} \left(1 - \frac{z_i}{Z} \right)^{5/6} \Delta z_i \quad (4.4)$$

$$C_n^2 = \alpha_T^2 \varepsilon^{-1/3} \chi_T \quad (4.5)$$

When designing the simulation, there are a number of other parameters that need to be controlled and changed depending on the environmental conditions, as well as other constraints that must be met to produce an accurate simulation. The primary parameters are the simulation resolution N , the side length of the source and observation planes D , grid spacing δ , the propagation distance between screens Δz , the wavelength of the light λ , the beam waist radius ω_0 , and the number of phase screens along the path of propagation n . Additionally, a value for c must be selected, which is a constant that represents the model's sensitivity to turbulence. Typically, $2 \leq c \leq 8$, where a value of 2 captures $\sim 97\%$ of the light and a value of 8 captures $\sim 99\%$ [28]. Selection of these parameters is informed by the three inequalities given in Eq. 4.6, 4.7. and 4.8.

$$(1) \quad \delta_n \leq \frac{\lambda \Delta z - (D + c \frac{\lambda \Delta z}{r_0}) \delta_1}{(D + c \frac{\lambda \Delta z}{r_0})} \quad (4.6)$$

$$(2) \quad N \geq \frac{(D + c \frac{\lambda \Delta z}{r_0})}{2\delta_1} + \frac{(D + c \frac{\lambda \Delta z}{r_0})}{2\delta_n} + \frac{\lambda \Delta z}{2\delta_1 \delta_n} \quad (4.7)$$

$$(3) \quad 2\delta_1 - \frac{\lambda \Delta z}{D} \leq \delta_n \leq 2\delta_1 + \frac{\lambda \Delta z}{D} \quad (4.8)$$

To use these constraints, it is also necessary to know the spatial coherence diameter r_0 , commonly known as the Fried parameter, for the spherical wave propagating through the complex environment. The spatial coherence radius ρ_0 of a spherical wave propagating in oceanic turbulence, given in Eq. 4.9, is a function of the wave number k , propagation distance Z , rate of dissipation of kinetic energy ε , rate of dissipation of mean-squared temperature χ_T , and ratio of temperature and salinity contribution ω [30]. The spatial coherence radius of a spherical wave propagating in oceanic turbulence is related to the spatial coherence diameter by Eq. 4.10 [31].

$$\rho_0 = [3.603 \times 10^{-7} k^2 (Z) \varepsilon^{-\frac{1}{3}} \frac{\chi_T}{2\omega^2} (0.419\omega^2 - 0.838\omega + 0.419)]^{-3/5} \quad (4.9)$$

$$r_0 = 2.1\rho_0 \quad (4.10)$$

The final constraint to meet is the minimum number of screens needed, related to the maximum distance between screens, Δz_{max} . This is given in Eq. 5.11, and is a function of the grid spacing δ_n , simulation resolution N , and wavelength λ . As a result of Eq. 5.32, the minimum number of screens is given in Eq. 5.12 [28].

$$\Delta z_{max} = \frac{(\delta_n)^2 N}{\lambda} \quad (4.11)$$

$$n_{min} = \text{ceil}\left(\frac{Z}{\Delta z_{max}}\right) + 1 \quad (4.12)$$

Once all of these constraints are met, the simulation is accurately parameterized. However, to produce accurately estimated results, the thin phase screens placed along the path of propagation must be properly generated.

Thin phase screens are a discrete realization of a randomly-modeled turbulence. These means to generate a random thin phase screen, computer-generated random numbers must be transformed into a two-dimensional screen that has the same statistical properties as the modeled turbulence. This is most commonly done using a method based on the Fourier Transform (FT), where the numbers are drawn from a power spectrum representing the desired medium [28] [32]. For this work, the Nikishov spectrum is used because it models the oceanic environment, meaning that it accounts for changes to both the temperature and salinity of the water [33]. Using the method outlined by Korotkova in [29], the Nikishov power spectrum, given in Fig. 4.2, was individually plotted for just the temperature (Eq. 4.13) and just the salinity (Eq. 4.14). The full Nikishov power spectrum is given in Eq. 4.15. In these equations C and A represent constants, ε is the rate of dissipation of kinetic energy per unit mass of fluid, χ_n represents the rate of dissipation of mean-squared temperature and salinity, ω is the ratio of the temperature and salinity contribution (ranging -5 to 0, temperature to salinity), and η is the Kolmogorov inner-scale number.

$$\phi_T(k) = C_0 \varepsilon^{-1/3} \chi_T k^{-5/3} [1 + C_1 (k\eta)^{2/3}] e^{-A_T \delta} \quad (4.13)$$

$$\phi_S(k) = C_0 \varepsilon^{-1/3} \chi_S k^{-5/3} [1 + C_1 (k\eta)^{2/3}] e^{-A_S \delta} \quad (4.14)$$

$$\phi_n(k) = C_0 \chi_n \varepsilon^{-1/3} k^{-\frac{5}{3}} \left[1 + C_1 (k\eta)^{\frac{2}{3}} \right] \frac{(\omega^2 e^{-A_T \delta} + \omega^{-2} e^{-A_S \delta} - 2\omega e^{-A_{TS} \delta})}{\omega^2 - 2\omega + 1} \quad (4.15)$$

When the spectra given by Eq. 4.13 and Eq. 4.14 are normalized and plotted, the figure given in Fig. 4.5 is produced.

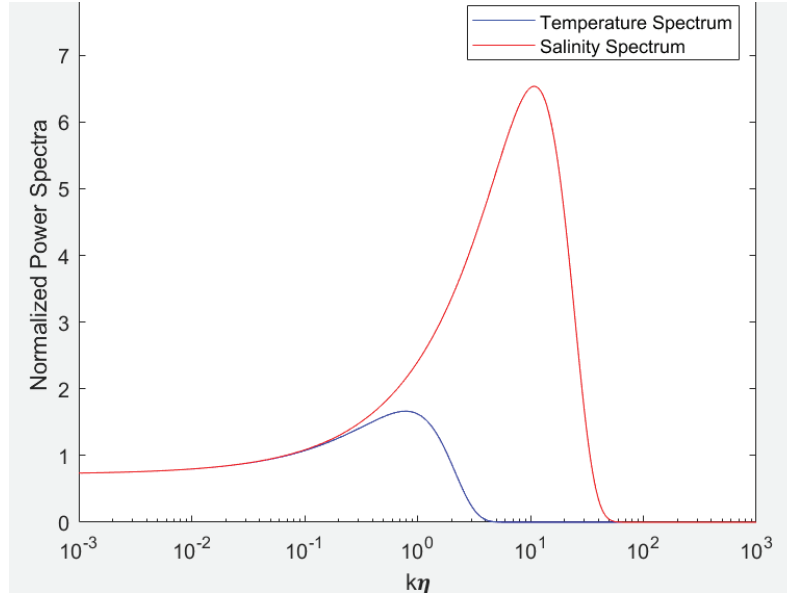


Figure 4.5: Normalized temperature and salinity Nikishov spectra [29]

From this power spectrum, a phase screen such as the one given in Fig. 4.6 is generated. This can be done using the provided function “ft_nikishovPS.m” given in Appendix A, Sect. 10.

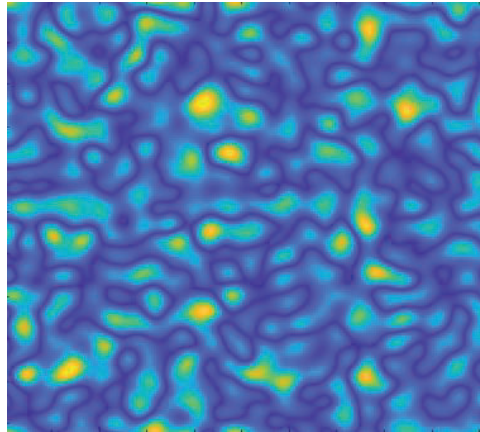


Figure 4.6: FT Nikishov Random Phase Screen

However, this screen alone does not produce results that match what is experimentally observed. This is because when the FT method is used to generate the phase screen, only the high spatial frequencies are captured, as the grid cannot be sampled with a high enough resolution to capture the low spatial frequencies. As can be seen in Fig. 4.5, there is significant power in the low spatial frequencies of the temperature-only Nikishov spectrum. To compensate, the subharmonic (SH) method is commonly used, combining random draws of Zernike polynomials with non-uniform sampling in the spatial-frequency domain. This is proposed in [34] and utilized by Schmidt [28]. The SH method produces screens such as the one given in Fig. 4.7, which stands in stark contrast to Fig. 4.6. A SH phase screen can be generated using “sh_nikishovPS.m” provided in Appendix A, Sect. 11.

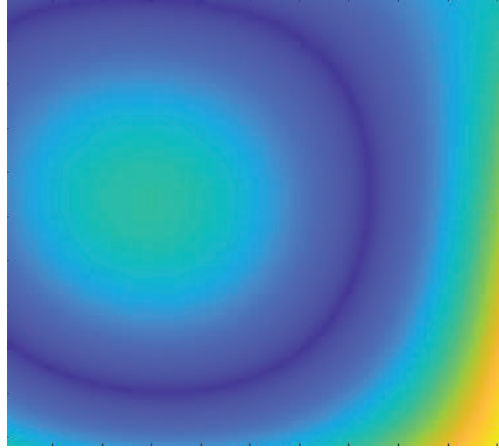


Figure 4.7: SH Nikishov Phase Screen

In order to create an accurate phase screen, the FT and SH phase screens can simply be added together, producing the phase screen show in Fig. 4.8. This can be seen in “ft_sh_nikishovPS.m”, provided in Appendix A, Sect. 12.

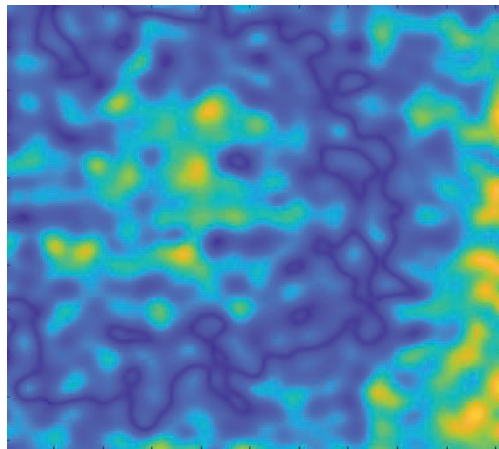


Figure 4.8: FT and SH Nikishov Phase Screen

When experimenting with phase screen generation, it was observed that the FT and SH phase screens have two different, but each well-documented, effect on the propagating beam. The FT screen, with power at the high spatial frequencies, causes the scintillation of the beam’s intensity, characterized by a shimmering of the irradiance of the light. Intuitively, it is clear how small-scale changes to the beam’s phase (Fig. 4.6) would create scintillation, as the shimmering is small-scale as well. The SH screen, with power at the low spatial frequencies, causes beam wander, pushing the beam around the receiver and causing the vortex of the LG beams to break, if strong enough. This too can be intuitively seen by the large-scale changes in phase. Both effects are typical to experimentally observe, and when the FT and SH screens are combined, both effects are observed in simulation. As an additional check for accuracy, 20 produced phase screens were correlated to check for proper randomization, and a low correlation of 0.411 was observed, shown in Fig. 4.9. The correlation analysis was completed in MATLAB using “Screen_Vali.m” provided in Appendix A, Sect. 13.

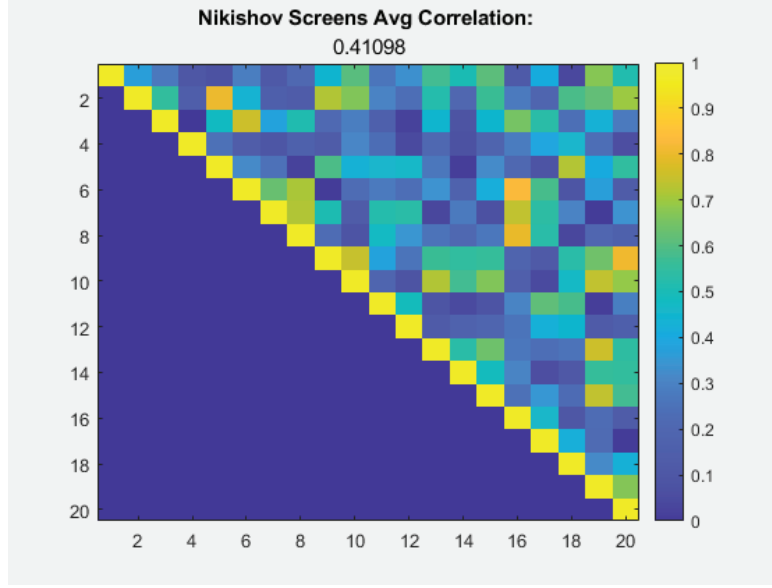


Figure 4.9: Correlation of 20 Nikishov Random Phase Screens

Before conducting a simulated evaluation of our beams, we studied the effect of changing simulation parameters to determine the optimal conditions and simulation parameters before the lengthy simulations were run. Depending on the resolution of the screens, and the number of realizations generated, the simulation can take 24-48 hours of continuous running to produce a full-sized, usable dataset. To examine the simulation, screens corresponding to three levels of turbulence were first generated, using $N = 1024$ and $D = 10$ cm. This is summarized in Table 4.2, with the screens in Fig. 4.10 corresponding to strong turbulence, Fig. 4.11 to moderate turbulence, and Fig. 4.12 to weak turbulence.

Qualitative Turbulence Level	$\varepsilon \left[\frac{m^2}{s^3} \right]$	$\chi_T \left[\frac{K^2}{s} \right]$
Strong (Surface Waters) [29]	1×10^{-2}	1×10^{-4}
Moderate [35]	1×10^{-5}	1×10^{-7}
Weak (Deep Waters) [29]	1×10^{-8}	1×10^{-10}

Table 4.2: Summary of Simulated Turbulence Levels and Parameters

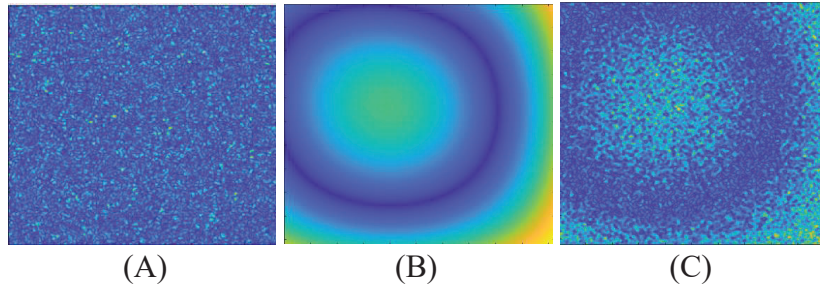


Figure 4.10: Strong turbulence phase screens (A) FT phase screen, (B) SH phase screen, and (C) combined phase screen

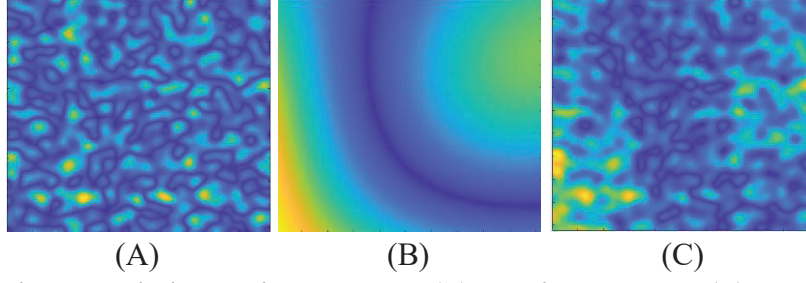


Figure 4.11: Moderate turbulence phase screens (A) FT phase screen, (B) SH phase screen, and (C) combined phase screen

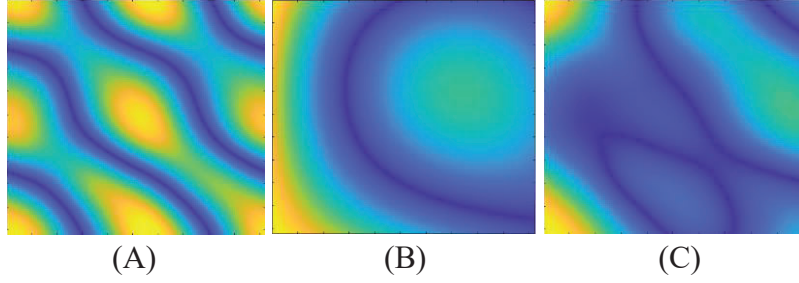


Figure 4.12: Weak turbulence phase screens (A) FT phase screen, (B) SH phase screen, and (C) combined phase screen

As is clear from Figures 4.10, 4.11, and 4.12, there is a significant difference in the generated FT phase screens, with very little difference in the SH phase screens. This describes how the high-spatial frequencies dominate the distortion of the light in strong turbulence while the low-spatial frequencies dominate in weak turbulence.

A single LG beam, with order 0 and topological charge 3, was then propagated at varying distances in all three levels of turbulence, using $N = 2048$, $D = 10$ cm, $n_{scrns} = 15$, and $\lambda = 633$ nm. The first propagation, through strong turbulence, is shown below in Fig. 4.13.

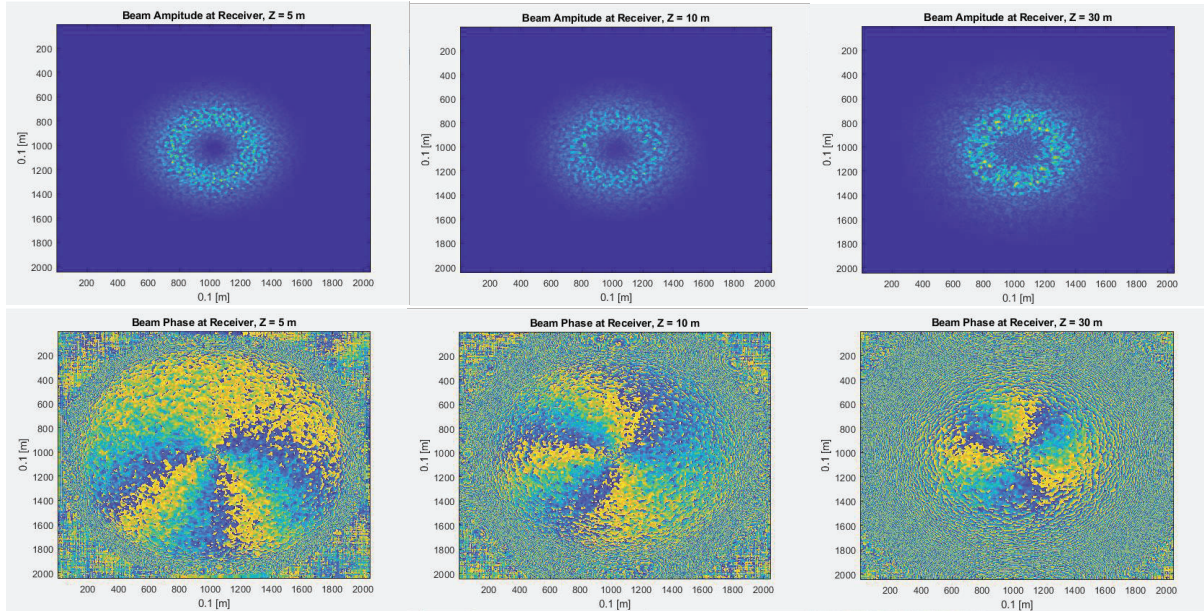


Figure 4.13: Beam $U_{0,3}$ in strong optical turbulence at distances $Z = 5, 10$, and 30 m

As the simulation is dominated by the scintillation, there is little utility in propagating a full alphabet through turbulence at this strength, as the beam quickly will become unrecognizable, to a human or the CNN.

The same LG beam was then propagated through moderate turbulence at varying distances, given in Fig. 4.14. This has more useful applications, because as can qualitatively be seen, the distortions are significant, but not overpowered by scintillation.

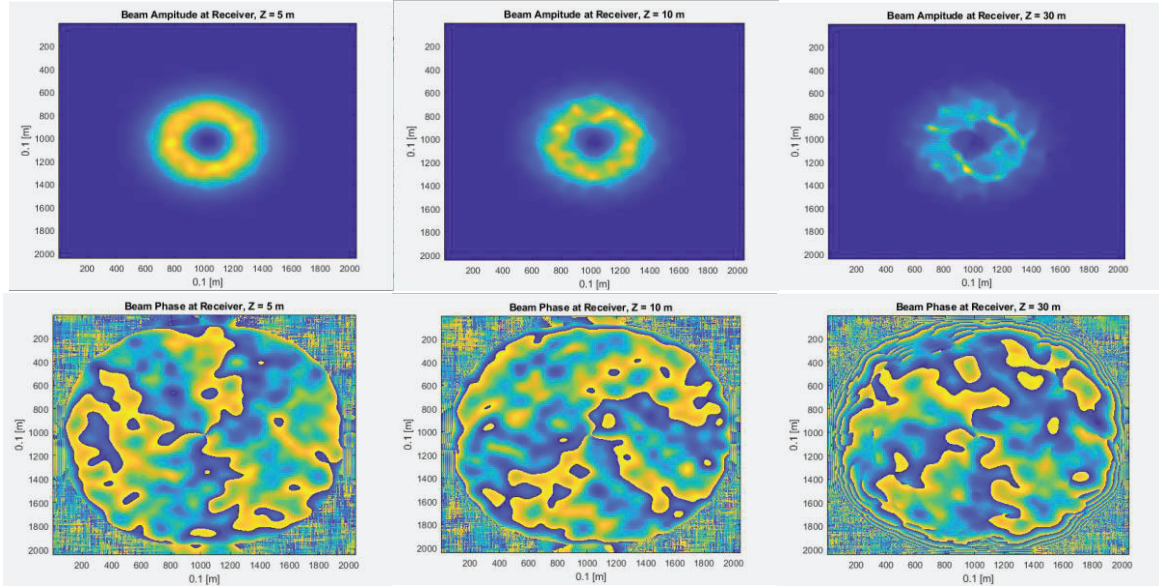


Figure 4.14: Beam $U_{0,3}$ in moderate optical turbulence at distances $Z = 5, 10$, and 30 m

Qualitatively, these results look very similar to what is experimentally observed in the lab, though in reality the propagation distance is much shorter. This is a byproduct of the weak fluctuation constraint of the split-step simulation.

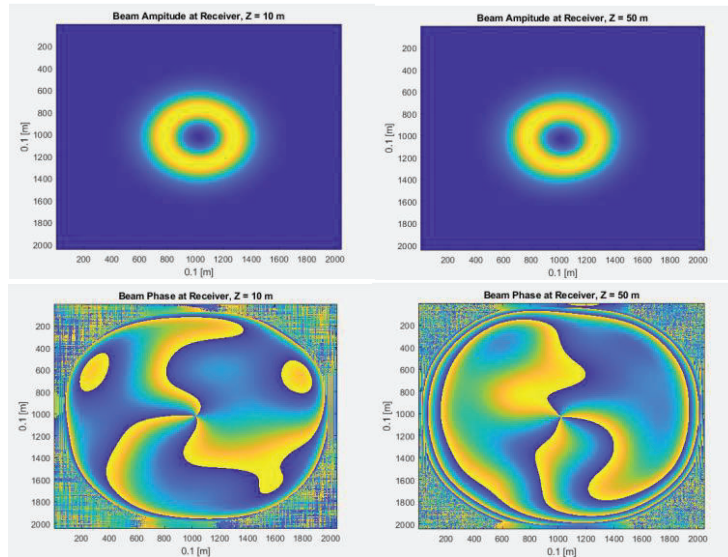


Figure 4.15: Beam $U_{0,3}$ in weak optical turbulence at distances $Z = 10$ and 50 m

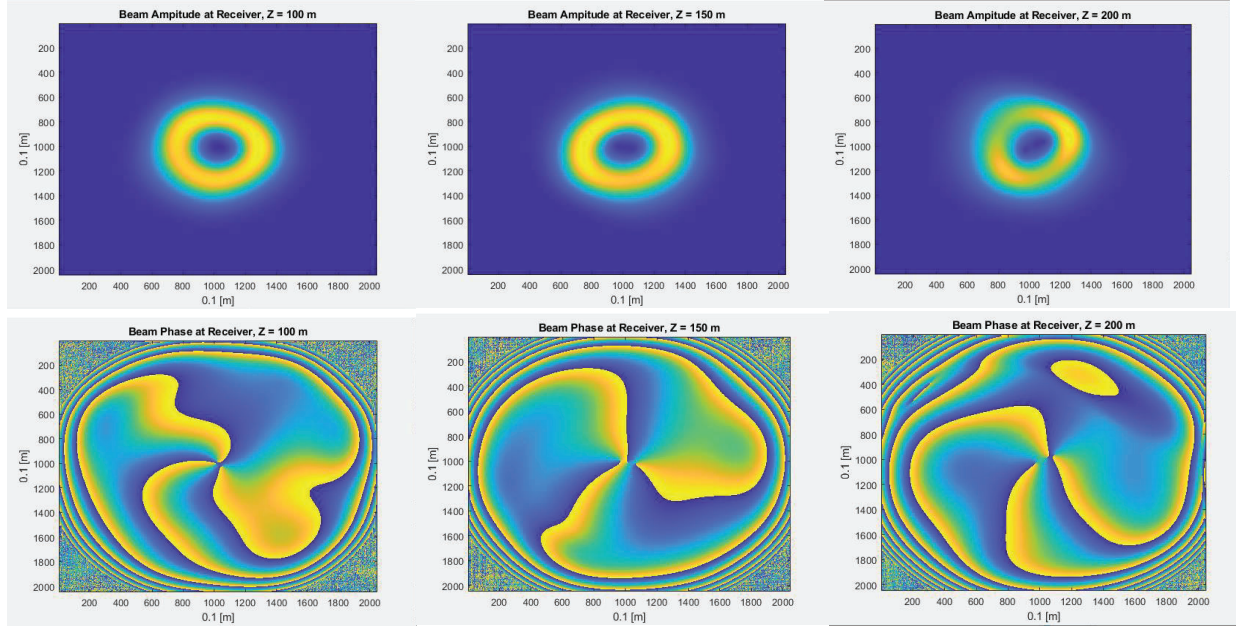


Figure 4.16: Beam $U_{0,3}$ in weak optical turbulence at distances $Z = 100, 150,$ and 200 m

In figures 4.15 and 4.16, the same LG beam is propagated through weak optical turbulence over varying distances. For this case, it is particularly interesting to examine the effect of the turbulence on the phase of the beam, and the impact this has on the vortex of the beam. In the case of $Z = 100$ m, the beam starts to become noticeably stretched, but the integrity of the vortex remains intact. At $Z = 150$ m, the singularity in the phase that corresponds with the vortex of the beam begins to break, corresponding to an ovaling of the beam and thinning of the ring that characterizes the intensity pattern. At $Z = 200$ m, the singularity is fully broken, which corresponds to the broken vortex in the intensity distribution. It has been shown that the CNN relies on vortex structure to accurately classify the beam [25]. Though propagating over unrealistic distances, the simulation still provides us with valuable insight into what causes the vortex of a beam to break. In order to compensate for distortions, it is important to better understand their root cause.

Section 4D: Hybrid Simulated and Experimental Set-Up

In order to mitigate the challenge of simulation constraints, a hybrid simulated and experimental environment was utilized. This was done by using two SLMs, one to generate the basis beams and one to represent the turbulent phase screen. This is possible as SLMs are designed to modulate the phase of the beam, allowing the simulated distortions to be experienced in the experimental environment by placing the generated phase screens used in simulation onto an SLM. This necessitates an experimental set-up with two SLMs, as shown in Fig. 4.17. The first SLM (SLM #1), 1920×1152 pixels, is used to generate the LG beams that are being evaluated through the simulated turbulence. The second SLM (SLM #2), 256×256 pixels, cycles through a large number of phase screens generated from the Nikishov spectrum in order to impact the incident beam. The phase screens utilized in the hybrid environment were generated at 256×256 pixels, as this was the size of our second SLM. This greatly sped up the phase screen generation process, as we were not constrained by the same parameters given in the simulation for generating screens of a certain resolution. For the experiments completed in the hybrid environment, 6000 screens were

produced using “genSLMTurbScrns.m” given in Appendix A, Sect. 9, and shown in Fig. 4.18. The distorted beams were then reflected and projected onto a screen, where they were captured by our high-speed camera. This also greatly sped up the data collection process, as the camera is able to collect and download data at a much faster rate than when using the optical wave split-step simulation. Each method of evaluation has its strengths and weaknesses, and all three were used to provide the most wholistic and rigorous evaluation process possible.

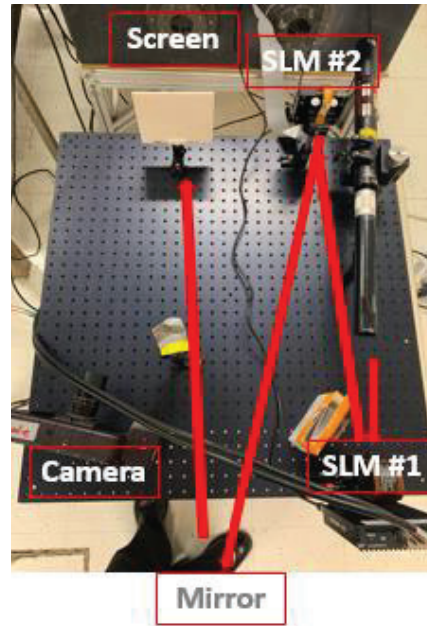
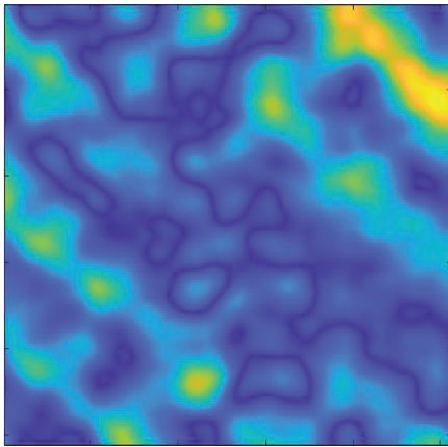
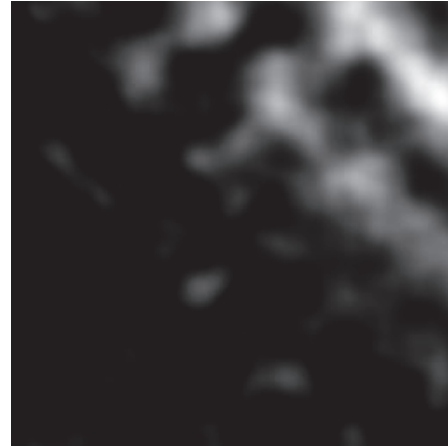


Figure 4.17: Hybrid simulation / experimental set-up



(A)



(B)

Figure 4.18: (A) Generated 256 x 256 Nikishov phase screen and resulting screen to place on SLM to provide simulated turbulence

Section 5: State of the Art in Published Literature

Section 5A: Resilience of Beams Carrying OAM Propagating through Complex Environments

In the field of UWOC, research into the use of light beams carrying OAM has grown over the last several years. Ren et al. [36] demonstrated in 2016 that the desired data transmission rates could be achieved using a modulated green laser carrying OAM. The orthogonality that is a result of the OAM allowed the beam to be spatially multiplexed, increasing the bit rate of the system. This proof of concept experiment demonstrated practically useful data transfer rates on the order of gigabits of data per second. The authors examined the propagation of the beams through various impacted mediums, looking at the impact of scattering particulate, mechanically induced currents, and thermally-generated optical turbulence. The authors used conjugate mode sorting to demultiplex the beams, but only qualitatively described the optical turbulence that affected the beams propagation. This study proved the practical utility of beams carrying OAM, but found that maintaining the necessary precise alignment for conjugate mode sorting was difficult in turbulence, and might require a pointing and tracking system. Other research has investigated the influence of turbulence on the propagation of beams carrying OAM. Cochenour et al. [37] examined the attenuation of beams carrying OAM in a multiple scattering environment. The authors report a higher intensity of signal as the topological charge of the beam carrying OAM increased. It is hypothesized that the same will be true for beams carrying OAM propagating through optical turbulence. This work, along with much more, laid the ground work to evaluate systems utilizing light carrying OAM.

Section 5B: Integrating CNN into UWOC Systems Utilizing Beams Carrying OAM

In 2017, Doster and Watnik [1] investigated the use of the resulting intensity patterns of spatially multiplexed beams carrying OAM to provide an alternative method of demultiplexing the information, using CNNs. This alternative solution was proposed in an effort to provide a solution to the challenges and complicated hardware that is required with the use of conjugate mode sorting. Doster and Watnik compared the accuracy, defined as percentage of beams classified accurately, of conjugate mode sorting and CNN-based demultiplexing. It was found that the CNN was both more accurate, and able to admit a larger encoding set, which ultimately would increase the overall data rate of the system. The study used the popular deep CNN named Alexnet [38], which won the ImageNet Classification Competition in 2012 and popularized the use of deep CNNs for image classification. The authors used 32 symbols in their alphabet, and simulated atmospheric turbulence using a split-step simulation and random phase screens, achieving classification rates of ~99.4%. While achieving high classification rates, the authors propagated their beams through simulated atmospheric turbulence, which weakly impacts the beams. This led future work to use custom CNNs, versus Alexnet, which were designed more specifically for classifying patterns.

Following this, Avramov-Zamurovic et al. [39] used a custom CNN based on one designed for the ImageNet Classification Challenge of 2014, which placed a specific emphasis on maintaining a reasonable computational budget, and maximizing utilization of network layers [40]. This CNN was used because a reasonable computational budget is important when building a high-bandwidth communication system. The authors transmitted 16-beam alphabet of LG beams, and collected data from 6 scenarios, which corresponded with either quiescent or turbid water, at an attenuation length of 4, 8, or 12, demonstrating high levels of classification were possible in worse

conditions when the CNN is custom developed and trained from scratch. The deep CNN was comprised of 15 layers, with the architecture being composed of three sets of a convolutional layer, a normalization layer, a rectified linear unit layer, and a max pooling layer. For each scenario, a new network using the same architecture was trained on the images collected from that specific scenario. Each network was then tested using images from each scenario, and it was found that the network trained on the “worst” images, which came from the turbid scenario with an attenuation length of 12, was the most robust and performed the best across each image set.

Lohani et al. [41] used an alphabet of 8 symbols distorted with simulated attenuation to achieve high accuracy rates. This work corroborated the work done by Doster and Watnik, providing more evidence for the strength of deep CNNs in solving this communication challenge. Further exploring the use of CNNs for demultiplexing of beams carrying OAM, Cui et al. [42] experimentally analyzed propagation in clear and mechanically stirred water, as well as using a thin phase screen simulation to study the effect of optical turbulence. The authors used 16 independent beams, a simple CNN composed of 7 layers, and propagated the beams through a 1-meter water tank. The authors found that higher-order OAM modes tended to be mis-classified more often and that increasing or decreasing the input pixel numbers had a significant impact on the classification accuracy. The authors suggested future research into using more robust deep CNNs and implementing adaptive optics to increase classification accuracy. Wang et al. [43] increased the alphabet size to 16 beams, and propagated the beams via split-step simulation through atmospheric turbulence. The authors use a custom 6-layer CNN, choosing to trade computational power for speed. The authors also provide evidence for the usefulness of superimposed OAM beams, finding coaxial OAM modes to maintain a greater resistance to turbulence than single OAM modes.

Additional work was completed by Neary et al. in 2020, where various custom CNNs were used to evaluate an alphabet of 32 symbols through the air, and 16 symbols through the water. Optical turbulence was induced using a random phase screen split-step simulation. This work helped to solidify the important features of a small, but strong CNN, and the number of layers necessary for high levels of classification [44]. Finally, Avramov-Zamurovic et al. induced thermally-generated turbulence in our in-lab tank using a heater, characterizing the level of turbulence using temperature measurements from placed thermocouples and validating via simulation. This work used an alphabet of 48 symbols, and achieved high classification rates even in the presence of refractive index fluctuations of a much higher magnitude than that seen in simulation [2].

The majority of work has been done in the simulated environment, which allows for controlled and more easily characterizable turbulence. The generation of characterizable turbulence is one of the greatest challenges in designing a system that will work in a practical environment. Your system may work in simulation, but might fail experimentally, when the turbulence is not homogenous and isotropic. Or in the case of our previous research, the CNN may be able to learn the distortions created by our heater, but these are not necessarily fully representative of natural optical turbulence found in the oceanic environment. Our work is compared to previous literature and presented in Table 5.

Paper	CNN	Alphabet Size	Medium	Optical Turbulence	Accuracy	Size of Test Dataset
Doster et al. 2017 [1]	AlexNet	32	Air	Simulated via random phase screen	99.3-99.4%	4,800
Lohani et al. 2018 [41]	Custom & VGG16	8	Air	None induced	94.08-100%	240
Cui et al. 2018 [42]	LeNet-5	8	Water	Simulated via random phase screen	90-100%	8000
Wang et al. 2019 [43]	Custom (6 layer)	16	Air	Simulated via random phase screen	93-100%	500
Neary et al. [44]	Various pre-trained networks	Air: 32 Water: 16	Air and Water	Air: simulated via random phase screen Water: none induced	88.1-100%	N/R
Avramov-Zamurovic et al. 2021 [2]	Custom	48	Water	Induced using in-water heater	93.67-99.93%	96,000
This Paper	Custom	256 and 1024	Water	Induced using in-water heater	96.16-97.09% (256) 91.26% (1024)	102,000

Table 5.1: Comparison of LG and CNN-based optical networks [2]

Section 5C: Statement of Novelty

The work presented in this report provides a novel contribution to the field of underwater wireless optical communications. First, we provide a comparison of light carrying OAM to other structured light, further validating our usage of LG beams in our communication system. Then we optimize our alphabet selection by creating a basis beam selection methodology, as the basis beams provide the backbone for symbol creation and data multiplexing. Following this, we validated our selection methodology and improved data transfer rate by increasing the size of our alphabet to 256 and 1024 symbols, allowing more bits of information to be multiplexed per message sent. We provide rigorous characterization of both simulated and experimental turbulence, using the scintillation of a Gaussian beam to estimate and quantify the turbulence in real-time. The magnitude of the refractive index structure constant is used to quantify the strength of turbulent fluctuations, optical turbulence, to be able to draw parallels between our simulated, hybrid and experimental environments. Furthermore, we explore additional challenges regarding system misalignment and the effect it has on CNN performance, discovering a significant reduction in classification accuracy. Finally, work was completed on the construction of a Rayleigh-Bénard turbulence tank which will allow better controlled and characterizable turbulence to be experimentally generated.

Section 6: Comparison of Structured Light for SPIE Remote Sensing 2021

As the deep CNNs used in this research only use the intensity pattern of the light at the receiver when classifying messages, they do not factor in how the light is formed and what properties the light contains. This provided a motivation for further evaluation of neural network classification of light, by way of a comparison between forms of structured light that carry highly diverse properties in Spring of 2021, which was submitted to the SPIE Remote Sensing 2021 conference. For this research, LG beams carrying OAM are compared to 2D projected Far Field Images (FFIs). The FFIs were utilized because they lack spatial coherence, are uncollimated, and do not carry OAM. Following acceptance of the research abstract in July of 2021, this project was finalized, published and presented [45]. The following sections detail the experiment and results.

Section 6A: Generation of Structured Light

Both types of structured light were generated using a 256 x 256 pixel phase-only spatial light modulator (SLM). The screens utilized on the SLM were generated using code from MATLAB. This code was modified from code utilized in experimental work using LG beams [13] and Gaussian Schell-model (GSM) beams [46]. Though the code to generate the GSM beams was used as the backbone for the generation of the FFIs, actual GSM beams were not used in this research.

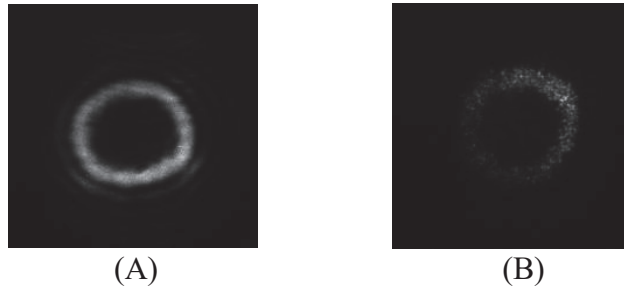


Figure 6.1: (A) LG beam at receiver and (B) FFI at receiver

Though similar at the receiver, as shown in Fig. 6.1, the SLM screens for the image generation are highly different for the LG beams and FFI. LG beams are coherent beams that carry OAM. This is in contrast to the FFIs, which are random and incoherent and lack phase information. The contrast between the two types of SLM screens can be seen in Fig. 6.2. These screens were each respectively used to generate the images shown in Fig. 6.1.

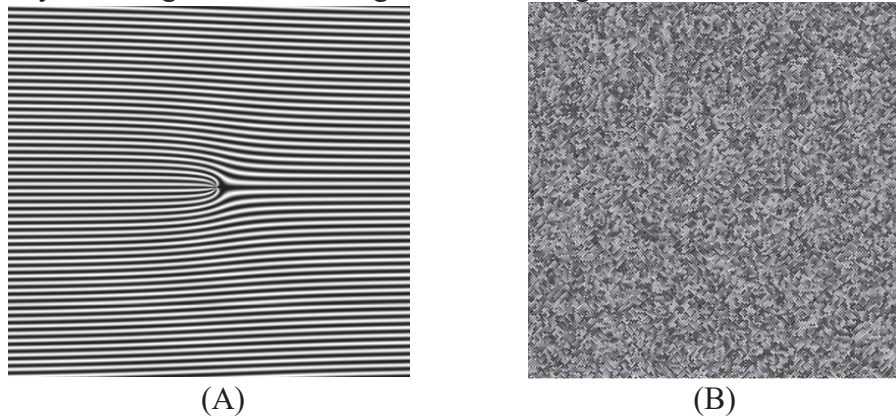


Figure 6.2: (A) LG beam SLM screen and (B) FFI SLM screen

The experimental work investigated three patterns for both the FFI images and LG beams, an example image of each is provided in Fig. 6.3



Figure 6.3: Example image of each pattern used in testing

Section 6B: Experimental Set-Up

For both the LG beams carrying OAM and the FFIs, the laser beam was propagated through a 2.5-meter tank of water and projected onto a screen, where it was then captured by our high-speed camera, which can be seen in Fig. 6.4. The camera captured images at 320×320 pixels, and with 8 bits and 2x binning. This was done to keep the size of the images down for faster processing. The camera operated at 50 FPS and with a 19-millisecond shutter. 2500 images of each beam type were collected, accounting for 50 seconds of propagation through turbulence per collection.

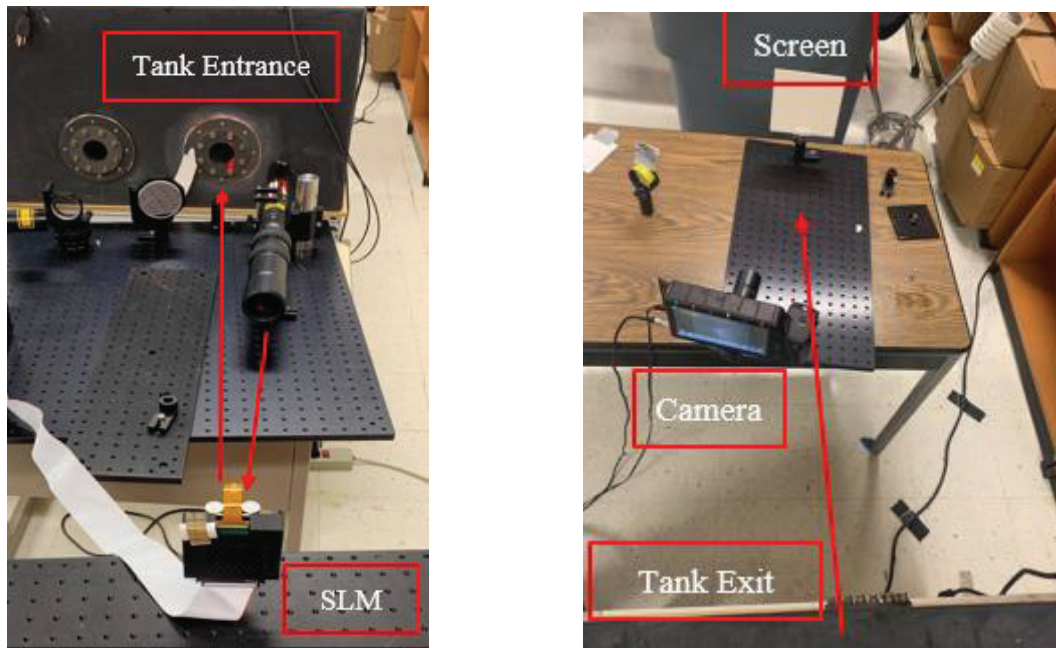


Figure 6.4: Experimental set-up

Originally, both types of structured light were directly propagated through turbulence, but in preliminary testing, it was found that the relative scale size of the turbulent eddies creating the optical turbulence and the beam width of the transmitted image had significant impact on the distortions seen in the light at the receiver. It was experimentally observed that the propagating light of the FFIs diverged at a much higher rate than the LG beams carrying OAM. This resulted

in a large discrepancy in beam width at the point of turbulence, where the LG beam had a diameter of 0.7 centimeters and the FFI had a diameter of 1.5 centimeters, shown in Fig. 6.5.



Figure 6.5: (A) LG beam (0.7 cm diameter) and (B) FFI (1.5 cm diameter)

This was further validated by using a shadowgraph to characterize the turbulent flow of the heater used to create the thermal fluctuations that result in the refractive-index fluctuations. As can be seen in Fig. 6.6, the turbulent flow is on the scale size of approximately 1 centimeter. In this, an important assumption was made being that the temperature fluctuation profile followed the velocity fluctuation profile, which is what can be qualitatively observed in the shadowgraph. A more accurate measurement would account for diffusion, but for the purpose of scale-size estimation, the velocity profile is deemed sufficient.

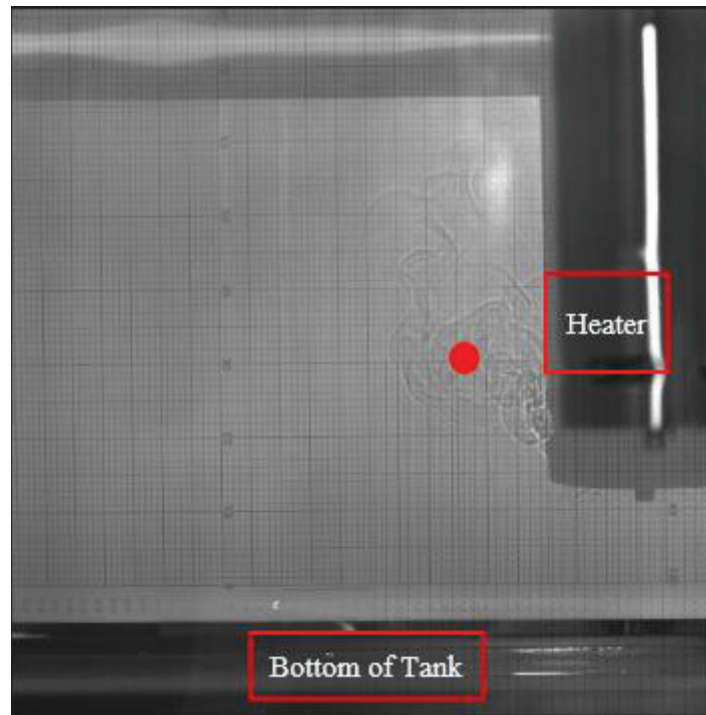


Figure 6.6: Heater turbulence characterization with approximate beam/image location shown

In order to correct this deficiency, a 4F lens system was utilized to partially-collimate and reduce in size the FFI. The lens system was comprised of a 400-millimeter lens and 300-millimeter lens separated by approximately 1 meter, shown in Fig. 6.7.

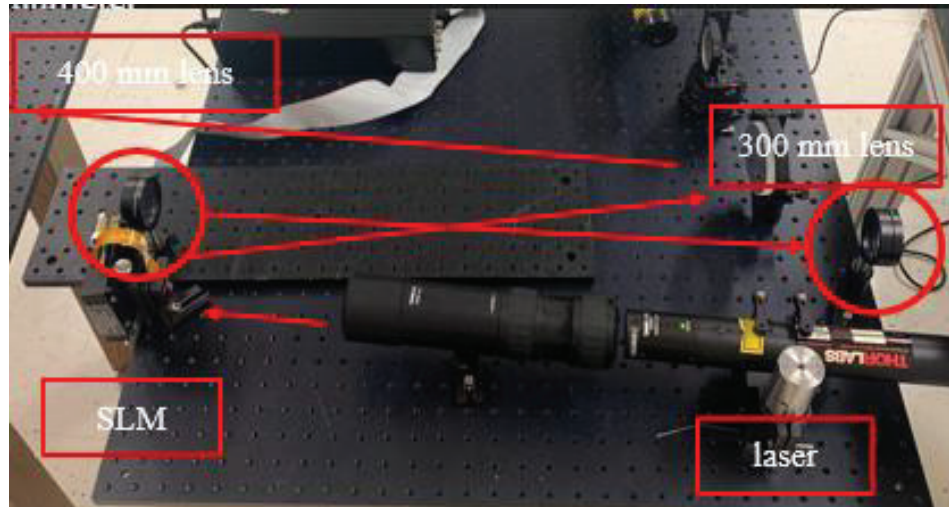


Figure 6.7: Added optical system for FFI to correct size discrepancy at turbulent point

Using the 4F lens system, the FFI at the turbulent point was decreased to 1-centimeter in diameter creating a much more similar turbulent effect on both the FFIs and LG beams, which were 0.7 centimeters in diameter (Fig. 6.4). The new FFI is shown in Fig. 6.8.

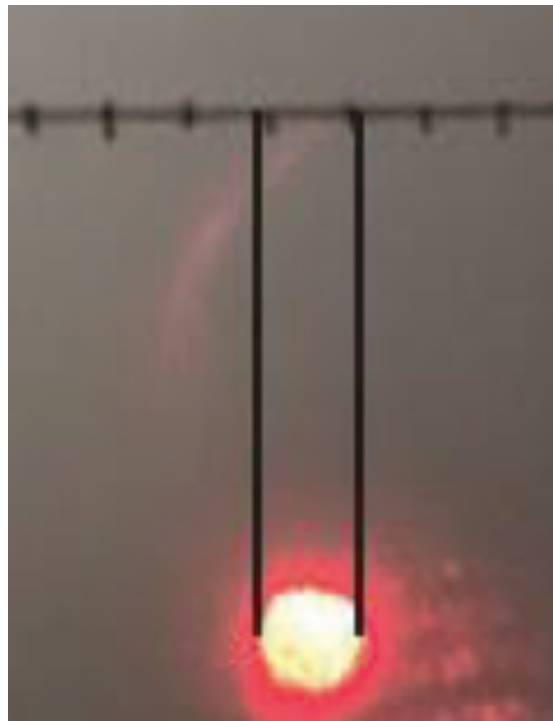


Figure 6.8: New FFI at turbulent point (1 cm diameter)

Section 6C: Use of Deep Convolutional Neural Network

As discussed in Sect. 3D, Convolutional Neural Networks (CNNs) provide an effective and reliable method for classifying images of beams carrying OAM, even in high attenuation and optical turbulence. For this investigation, the CNN architecture given in Sect. 3B, Fig. 3.2 was used. This particular architecture consists of an input layer, three sets of 2D-convolutional, rectified linear unit, and max pooling layers, and finishes with a 3-node fully connected layer. The deep CNN was designed, trained, and used to test the images using MATLAB 2021a and the Deep Learning Toolbox. The images were preprocessed using a bicubic interpolation to resize the images from 320 x 320 pixels to 64 x 64 pixels. The images were reduced in size to speed up the training and classification speed of the CNN, an important consideration for a communication system. The CNN is only comprised of 15 layers for the same reason, as larger CNNs are generally more computationally complex. The images were randomly assigned to the training, validation, and testing sets, with 40% being used for training, 50% being used for testing, and 10% used for validation. Training used the stochastic gradient descent method with momentum of 0.9, with an initial learn rate of 0.01 and mini batch size of 64. The training was terminated after a maximum of 10 failed validation checks.

Section 6D: Results and Conclusion

After propagation through the optical turbulence, there was a large qualitative difference between how the turbulence affected the LG beams and FFIs. It is well-documented that the LG beams distort through stretching, intensity scintillation, and movement of the beam vortex [2] [17]. This was reaffirmed through our own study. Interestingly, this was markedly in contrast with the way that the FFIs were affected by the changes to the refractive-index. Rather than shimmering and beam movement (wander), the FFIs appears to disintegrate and reform as the optical turbulence passed through. The static difference between these affects can be seen in Fig. 6.9.



Figure 6.9: (a) FFI partially disintegrated by OT and (b) LG beam distorted by OT

2500 images of three patterns (shown previously in Fig. 6.3) were collected for both the LG beams and FFIs, resulting in 15,000 total images. Two CNNs were trained, with one being solely on the images of the LG beams and the other on the FFIs. 1000 images of each pattern were used to train each CNN (3000 images total), and the remaining 4500 images were used in validation and testing. It was found that there was a 99.56% accuracy for the LG beam CNN and a 98.84% accuracy for the FFI CNN. The breakdown of which patterns were mistaken for other patterns can be seen in Fig. 4.10 and Fig. 4.11. From these results two important conclusions can be drawn. First, the CNN appears to prefer the type of distortion seen in the LG beams over the type of distortion of the FFIs. This is potentially due to the fact the CNN is able to learn the stretching, scintillation,

and wander of the LG beams, but cannot learn the disintegration of the FFIs as well. This potential explanation is strengthened by the confusion chart for both. For the LG beams (Fig. 6.10), the misclassifications consistently happened between two patterns (pattern 1 and pattern 3). This is in contrast to the FFIs (Fig. 6.11), where misclassifications occurred between patterns 1 and 2, and then also patterns 2 and 3.

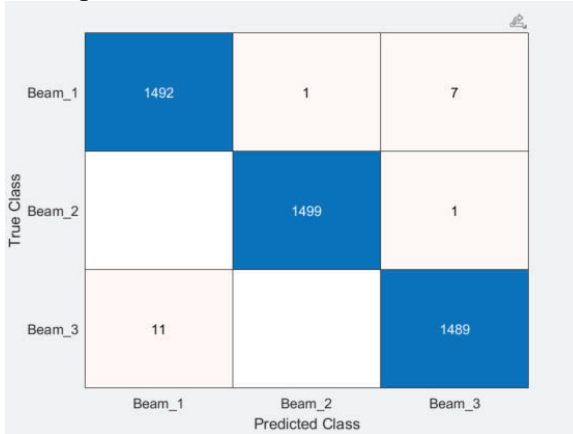


Figure 6.10: LG beam confusion chart

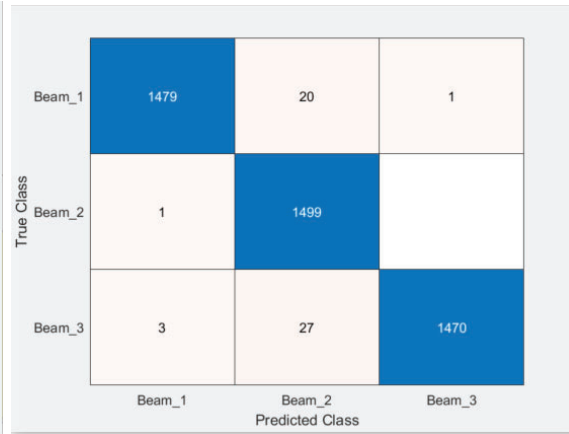


Figure 6.11: FFI confusion chart

However, it is important to note that not enough data nor enough patterns were compared to be able to definitively say for sure which the CNN “prefers”. Both the LG beam CNN and FFI CNN classified the images with very high accuracies, which provides further evidence for use of machine learning-based classification systems to be utilized in communication systems to decrease error rates.

Section 7: Basis Beam Selection

As discussed at the start of this report, the data transfer rate of this UWOC system can be increased by growing the alphabet used to transmit information. However, a system is only practical if it can be reliably classified in realistic conditions. The first evaluation we completed was to select for the ideal basis beams, which form the basis of the alphabet. Our hypothesis was that the informed selection of our basis beams would create a more robust and resilient network as we explore 256-beam and 1024-beam alphabets. This was done in the simulated, hybrid, and experimental environments, evaluating 36 basis beam “candidates” ranging from a topological charge of 1-9, and order of the Laguerre polynomial 0-3, as shown in Fig. 7.1. This candidate set was chosen to meet size constraints on our SLM screen. The premise of the evaluation was to propagate the beams through various turbulent environments and scenarios, and to identify trends related to the topological charge and order of the beams.

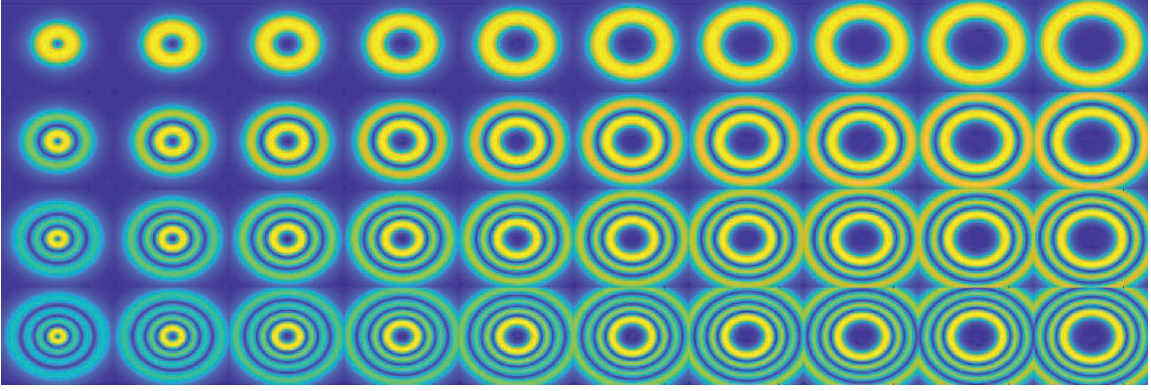


Figure 7.1: Theoretical images of basis beam candidates. From top to bottom, order of Laguerre polynomial (n) increases from 0 to 3. From left to right, topological charge (m) increases from 1 to 9. The images in the top left corner and bottom right corner corresponds to $U_{0,1}$ and $U_{3,9}$ respectively.

Section 7A: Evaluation through Simulation

Beginning with the split-step simulation, the basis beam candidates were evaluated through two levels of simulated optical turbulence, with parameters $N = 1024$, $D = 10$ cm, $Z = 30$ m, $n_{scrns} = 10$, and $\lambda = 633$ nm. For Case A (weaker), $\varepsilon = 1 \times 10^{-6} \left[\frac{m^2}{s^3} \right]$ and $\chi_T = 1 \times 10^{-7} \left[\frac{K^2}{s} \right]$. For Case B (stronger), $\varepsilon = 1 \times 10^{-5} \left[\frac{m^2}{s^3} \right]$ and $\chi_T = 1 \times 10^{-7} \left[\frac{K^2}{s} \right]$. For the simulated experiment, 1000 images of each beam were collected for Case A. 100 images of each beam were used to train the simulation, and the remaining 900 images were used to test the CNN. It was found that in this level of turbulence, it was simple for the CNN to learn the differences and classify accordingly, **as a classification accuracy of 100% was achieved**. This did not provide evidence towards which basis beams should be utilized, it does provide evidence for the power of CNNs to learn small differences, in this case vortex radii, and classify accordingly.

For Case B, the turbulence was increased to try and “break” the CNN. Again, the CNN had no issue classifying the basis beams with 100% accuracy. For Case B, 500 images of each beam were collected, using 100 images to train the CNN and the remaining images for testing. The number of images collected was cut in half to decrease the simulation runtime. The CNN used for Case A and Case B was the same CNN given in Sect. 3B, Fig. 3.2, with the same training parameters discussed in Sect. 6C [17]. Examples of the distortions from turbulence from these experiments are given below in Fig. 7.2, and the results from this experiment are summarized in Table 7.1.

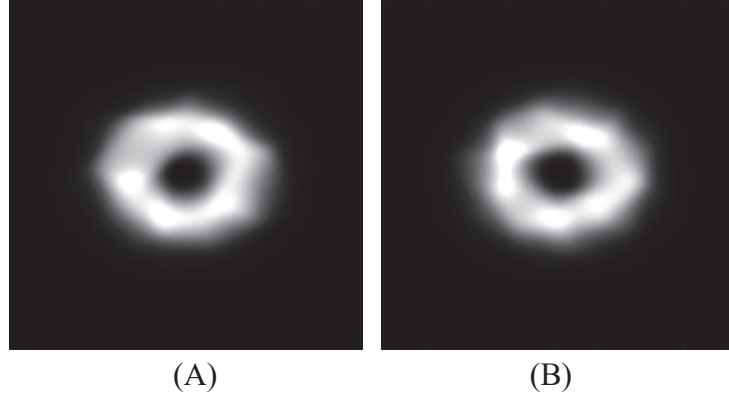


Figure 7.2: Example image of Beam $U_{0,3}$ distorted by optical turbulence in (A) Case A and (B) Case B

The split-step optical wave simulation is only valid in the weak turbulence regime. This is a demonstrated weakness of the simulation, as beams must be propagated over unrealistic distances in order to see observed effects in experimental strong turbulence. Weak fluctuations are easily learned by the CNN, indicating their robust ability to adapt to small environmental changes. Additionally, only a limited dataset could be collected due to the processing time and power it required to produce large datasets via simulation.

Case	$C_n^2 [m^{-\frac{2}{3}}]$ [29]	CNN Accuracy
A – simulation (100 training images)	3.14×10^{-13}	100%
B – simulation (100 training images)	6.76×10^{-13}	100%

Table 7.1: CNN classification accuracies relative to refractive-index structure constant values for basis beam evaluation using a split-step optical wave simulation

Section 7B: Evaluation through Hybrid Set-Up

In order to mitigate the challenge of creating a large dataset, our hybrid, simulated / experimental environment was utilized. Case A and Case B from the previous experiment were repeated, with the simulated turbulence being placed on the 256 x 256-pixel SLM. Given the decreased resolution of the turbulent phase screens, 6000 unique screens for each case were generated. The screens were generated using the same process as for the simulation, and therefore were created with the same level of turbulence for Case A and Case B, allowing for a one-to-one comparison. The screens were cycled at a rate of 200 Hz (every 5 ms). The SLM generating the basis beams was cycled at a rate of 20 Hz (every 50 ms), and the camera collecting the images operated at 275 FPS. For each cycle of the basis beam, 11 images were captured (40 ms total). The 11th frame was then dropped in post-processing to minimize synchronization error between the camera and SLM as previously discussed. This resulted in a total collection time of 185 seconds and 1100 images per beam, with 1000 usable images. In using two SLMs for this experiment, we found significant and unwanted optical effects occurred. The second SLM with

the smaller screen (256 x 256) resulted in a squaring of the beam, more prominent at the lower orders. Fig. 7.3 demonstrates the difference between LG beams $U_{1,1}$ and $U_{1,9}$.

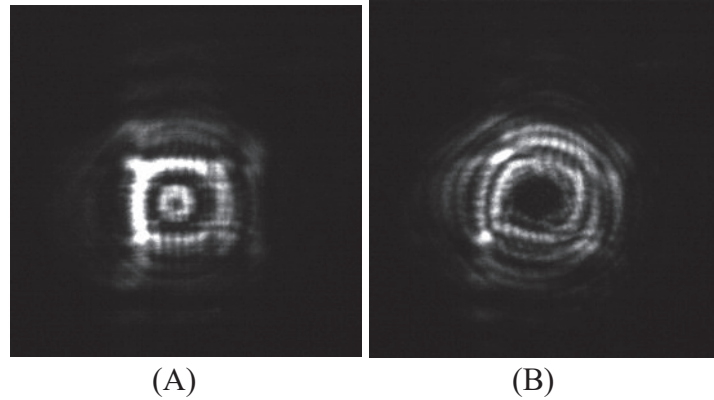


Figure 7.3: Example images of (A) Beam $U_{1,1}$ and (B) Beam $U_{1,9}$ in weak turbulence highlighting the difference in squaring in the hybrid simulated and experimental environment

Much like in the simulated environment, there was little difference between the weak (Case A) and strong (Case B) environments. In Fig. 7.4, there is a contrast between the same beam ($U_{0,3}$) for both cases. Case B appears to have slightly more distortion, but as can be seen by Table 7.2, the classification accuracy is practically the same, indicating the CNN had little issues with the weak fluctuations. It is noteworthy that the CNN performed worse for the hybrid environment for the same level of generated turbulence, due to the artifacts, and degradation of beam quality that occurs in the physical world, and is not present in the simulation. This can be seen by comparing Fig 7.2 with Fig. 7.4. This degradation resulted in the beams being misclassified for their adjacent topological charges in all except 5 cases in Case A (Fig. 7.5) and 2 cases in Case B (Fig. 7.6). The below confusion matrices provide this information and highlight where the misclassification trends occurred. In Fig. 7.7 we see the inaccuracy graph for Case A, where there are no clear trends regarding which beams are “worse” than others. This is true for Fig. 7.8 as well, though we see a higher level of misclassification for beams $U_{0,5} - U_{0,8}$ indicating the CNN may have had trouble distinguishing between the beams at lower versus higher orders.

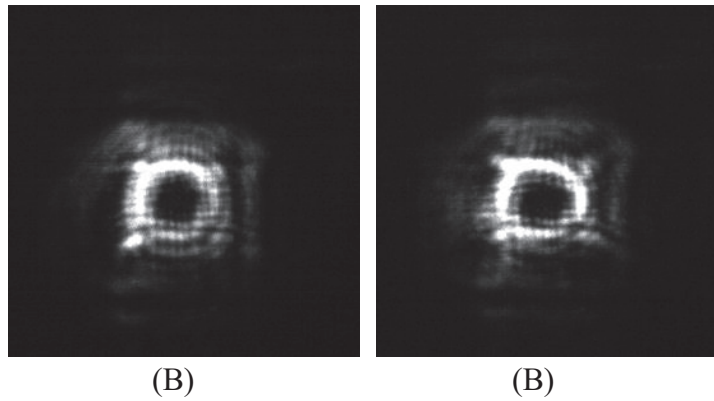


Figure 7.4: Example images of Beam $U_{0,3}$ distorted by optical turbulence in (A) Case A and (B) Case B

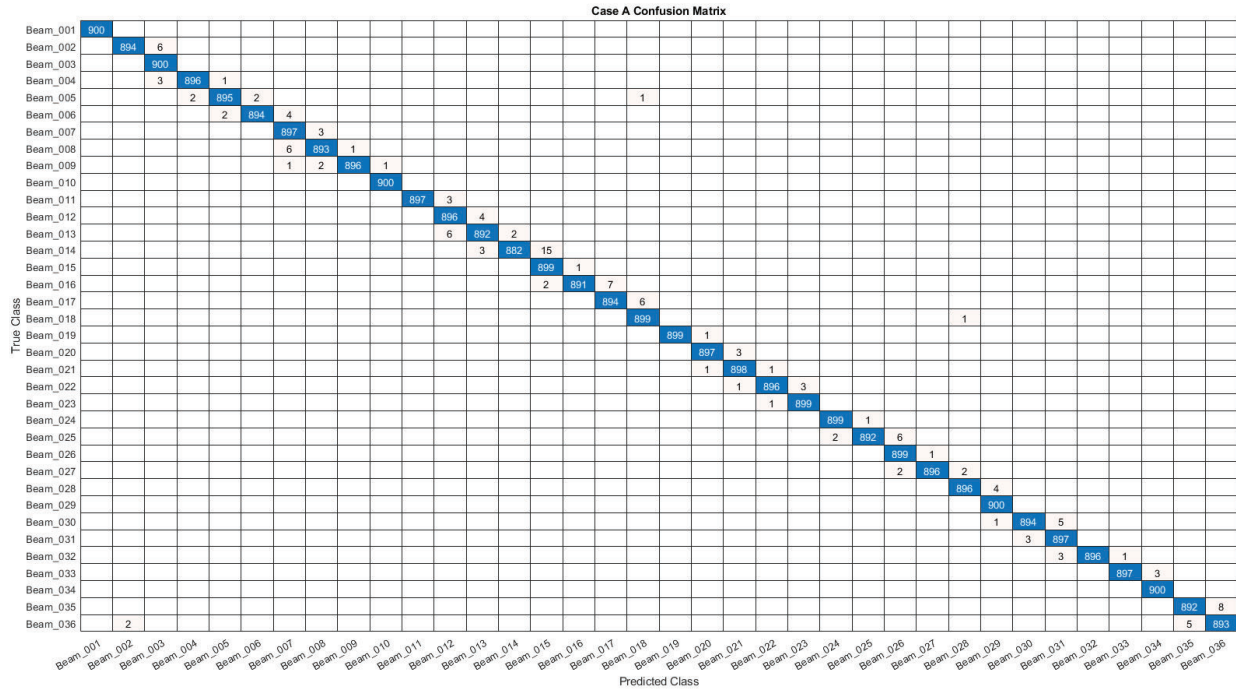


Figure 7.5: Confusion Matrix for Case A indicating what every beam was misclassified as

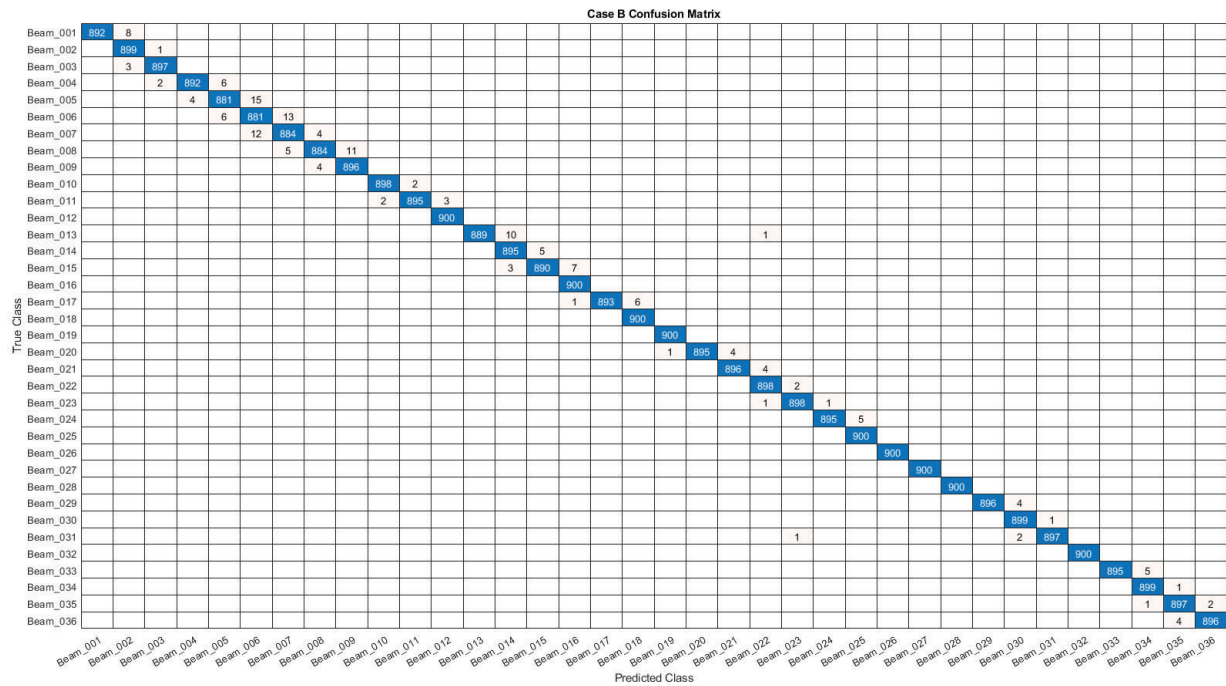




Figure 7.7: Inaccuracy graph for Case A demonstrating no clear trends in the misclassifications of specific beams

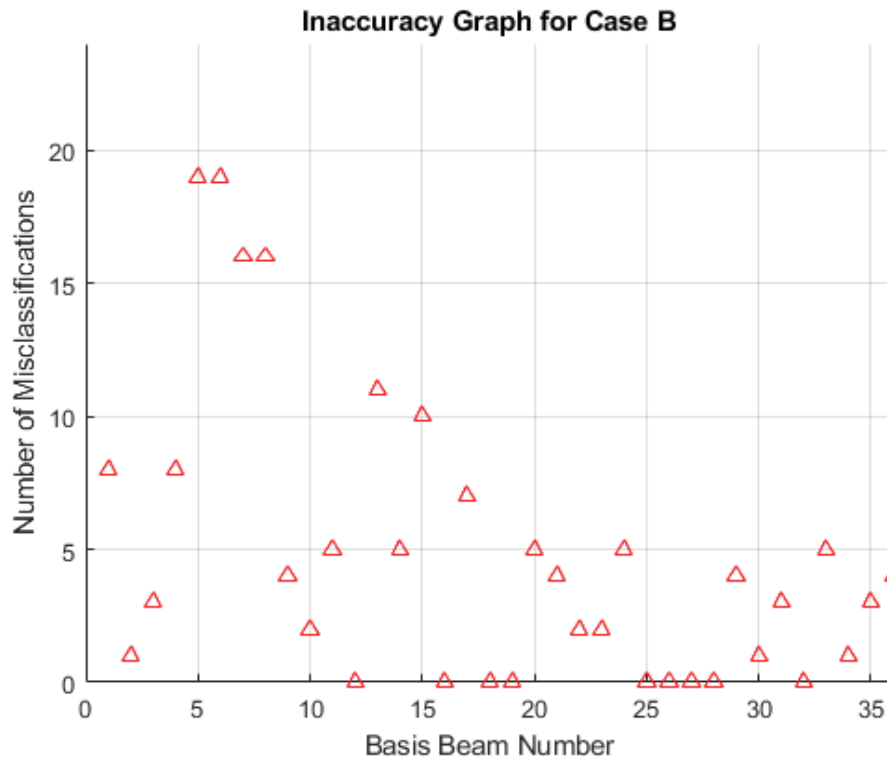


Figure 7.8: Inaccuracy graph for Case A demonstrating a small trend in the misclassifications of beams at lower orders

Case	$C_n^2 [m^{-\frac{2}{3}}]$ from Screens [29]	CNN Accuracy
A – simulation (100 training images)	3.14×10^{-13}	100%
B – simulation (100 training images)	6.76×10^{-13}	100%
A – hybrid (100 training images)	3.14×10^{-13}	99.55%
B – hybrid (100 training images)	6.76×10^{-13}	99.47%

Table 7.2: CNN classification accuracy for hybrid simulation / experiment

Section 7C: Experimental Evaluation of Basis Beams

Following the evaluation of the basis beams through the simulated and hybrid set-ups, the basis beams were evaluated using our experimental set-up. The 36 beams were cycled through 100 loops at a rate of 20 Hz, changing the beam every 50 ms. A Gaussian beam was placed at the start of every cycle for turbulence characterization. The camera captured images at 275 FPS, with a resolution of 320 x 320 and 2x binning. The camera captured 11 images per trigger, with the 11th image dropped in post processing. Given the controllability of variation in our experimental set-up, only one case of turbulence was used for the evaluation. In Case C, CNN was trained on 100 images like in Case A and Case B, and in Case D, the CNN was trained on 500 images in order to reach higher levels of accuracy. Sample images of the LG beam $U_{0,3}$ are given below in Fig. 7.9.

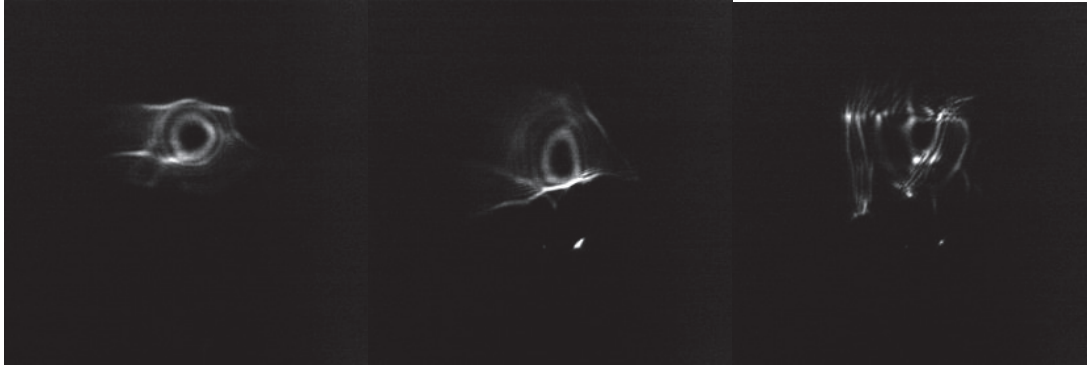


Figure 7.9: Several examples of Beam $U_{0,3}$ in the presence of optical turbulence. All images are from the same collection, demonstrating the variety and impact of the turbulence

The value for the refractive index structure constant was estimated using the method outlined in Sect. 3C. Fig. 7.10A provides the normalized and time-averaged intensity for the Gaussian beam placed in the basis beam experiment and Fig. 7.10B provides the pixel-by-pixel SI of the beam. The value for C_n^2 is recorded in Table 7.3, and a graph of the variations over time is provided in Fig. 7.11 to validate the averaging process. It can be seen that while the estimated C_n^2 varied frequently, it stayed within the same order of magnitude for the whole collection process. The optical artifact seen below the main Gaussian shape was likely the result of contamination on the

tank window, but had little to no impact on estimating the refractive index structure constant or CNN performance.

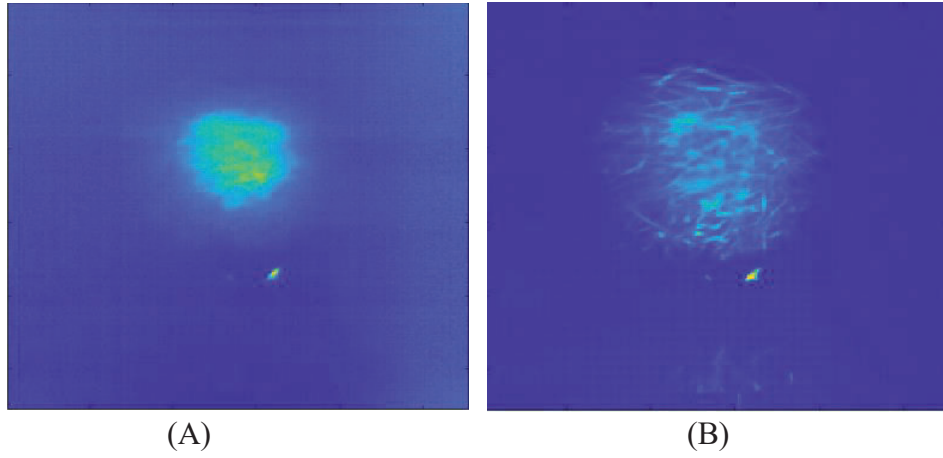


Figure 7.10: (A) Time-averaged an normalized intensity of Gaussian beam and (B) pixel-by-pixel time-averaged SI of Gaussian beam

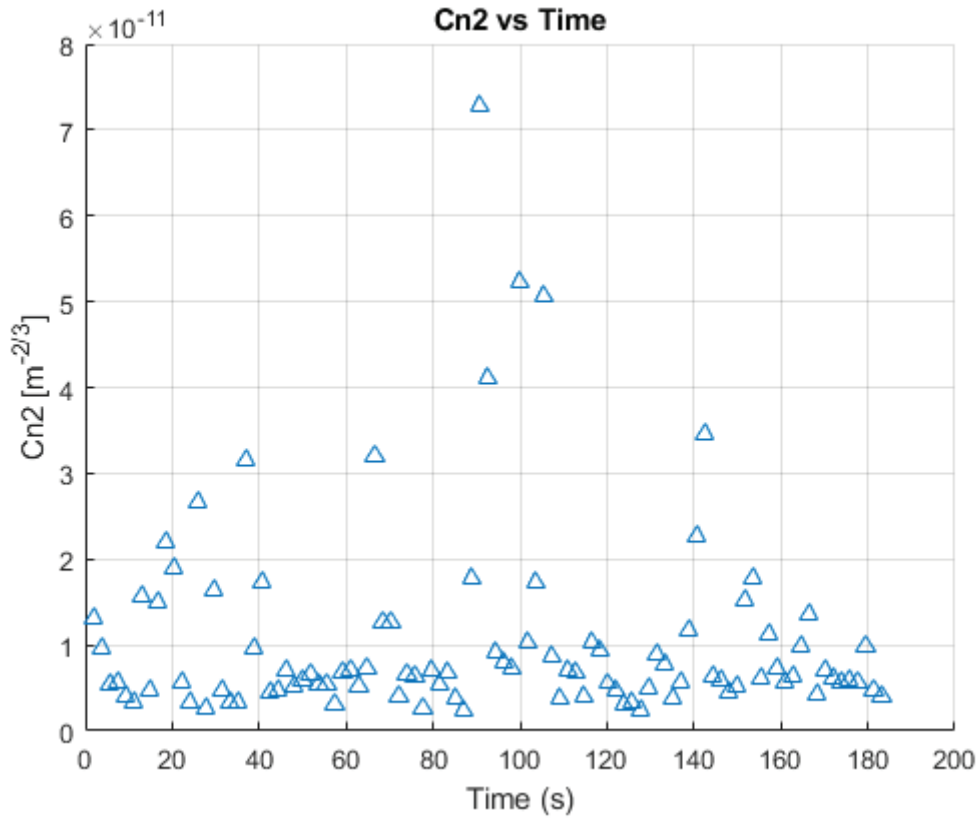


Figure 7.11: Time-varying values for C_n^2 taken as a time-average of every 10 frames, or every 1.85 seconds

Case	$C_n^2 [m^{-\frac{2}{3}} [29]$	CNN Accuracy
A – simulation (100 training images)	3.14×10^{-13}	100%
B – simulation (100 training images)	6.76×10^{-13}	100%
A – hybrid (100 training images)	3.14×10^{-13}	99.55%
B – hybrid (100 training images)	6.76×10^{-13}	99.47%
C – experimental (100 training images)	2.62×10^{-10}	60.40%
D – experimental (500 training images)	2.62×10^{-10}	98.46%

Table 7.3: Comparison of basis beam classification accuracy from Cases A-D

As can be seen, there is a significant drop off between Case B and Case C, as the experimentally induced turbulence has a much greater level of variation. This necessitates a larger training dataset to achieve comparable classification accuracies. Though the collection and training of larger datasets requires more time, in a practical setting this is only a minor issue because all of that can be done before a communication system goes operational. Case D is presented to provide a more accurate comparison, achieving a much higher accuracy on the CNN. As can be seen in Fig. 7.12, the misclassifications in Case C were concentrated along adjacent beams, and focused within similar Laguerre polynomial orders. This intuitively makes sense, as the change in the beam vortex is small and can be distorted by turbulence, causing many adjacent misclassifications, but having differing numbers of rings makes it a simpler for the CNN to distinguish between beams $U_{0,6}$ and $U_{1,6}$, for example. In Fig. 7.13, we present the confusion matrix Case D, which has a much lower number of misclassifications. However, it can still be seen that the misclassifications are concentrated along adjacent beams, located on the diagonal running through the matrix. Furthermore, Fig. 7.14 demonstrates no distinguishable trends for basis beam performance, indicating that selection of higher versus lower topological charges is not as important as having variety and diversity in alphabet selection.

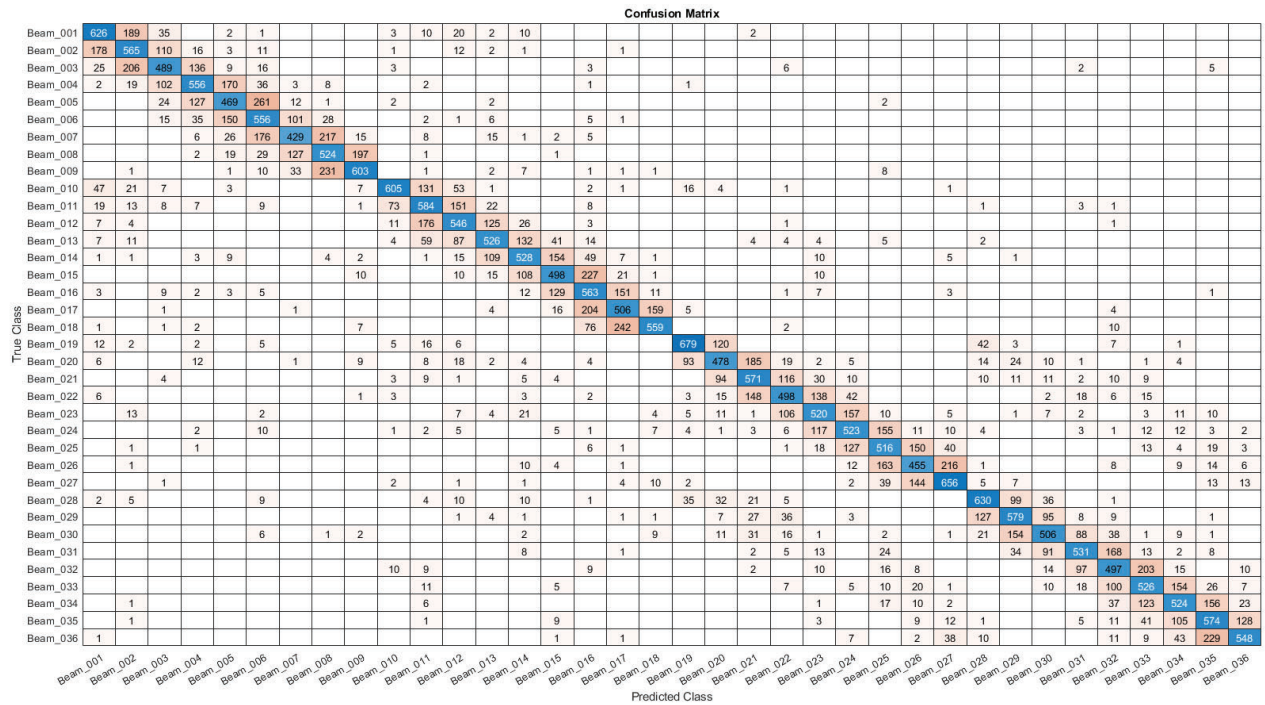
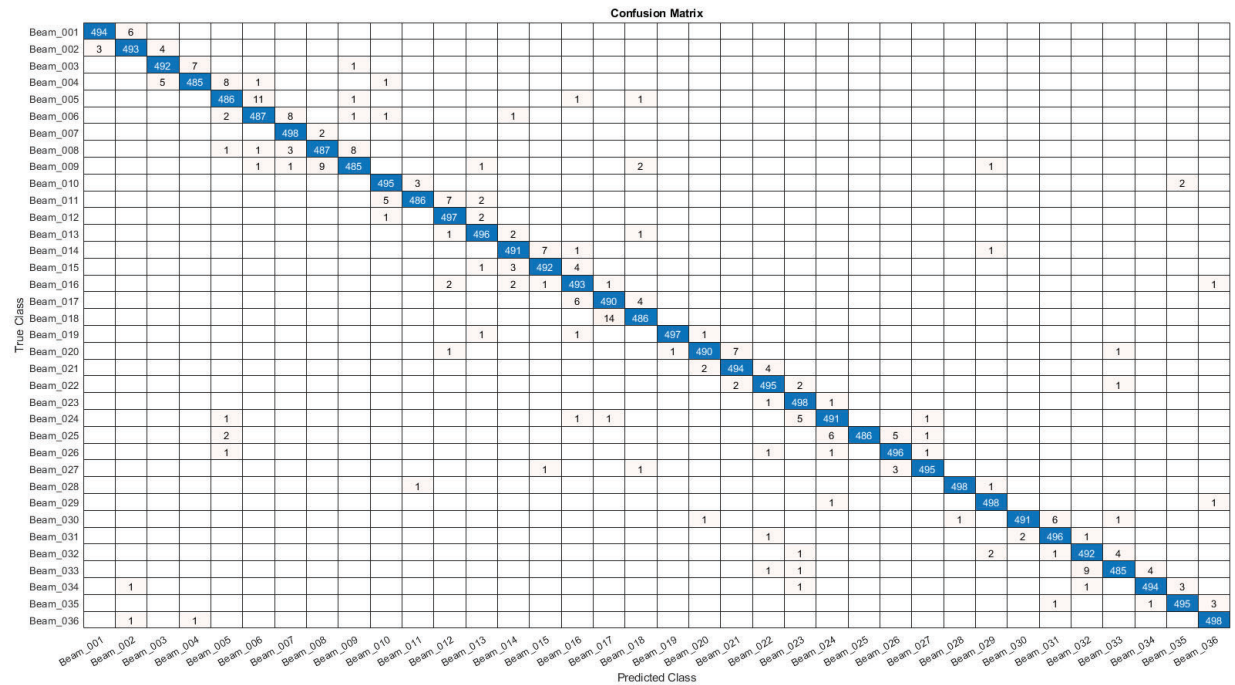


Figure 7.12: Confusion matrix for Case C demonstrating the high number of misclassifications and concentration of misclassifications along adjacent beams



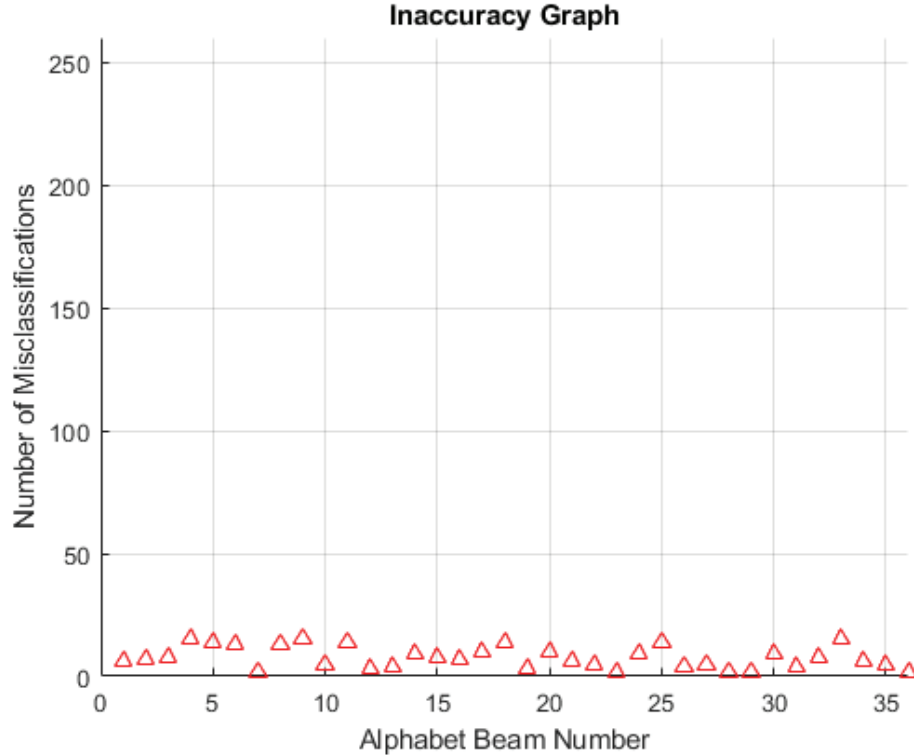


Figure 7.14: Inaccuracy graph for Case D indicating no distinguishable trends among misclassifications

Section 7D: Novel Basis Beam Selection Methodology

When designing system alphabets, it is not practical to select the basis beams on a trial and error basis, due to the number of changeable variables, and the labor required in testing and evaluating your network. Therefore, we develop a novel methodology for basis beam selection, using the basis beam evaluation listed previously, and previous research as our reference. First, we recommend starting with topological charges that are as high as feasible, as previous work suggests they are more resilient in the presence of optical turbulence [25]. Second, an equal amount of varying Laguerre polynomial orders should be used for maximum diversification of the alphabet. This research has shown no distinguishable preference for varying orders, but basis beams were primarily misclassified within their own Laguerre polynomial order. Third, adjacent topological charges, with the same order, should not be chosen to avoid similarities in both the basis beams and across the full alphabet. As discussed, the basis beam misclassifications were concentrated along beams with similar intensity patterns, therefore this should be avoided for optimal performance. Fourth, a negative topological charge should be added every 3rd basis beam to create “petal” structures in the resulting alphabet, which act as distinct, classifiable structures for the CNN [13]. Finally, the alphabet should be checked with a correlation matrix following generation, as this is a quick and easy way to ensure pattern diversification. This methodology is summarized below, and will be subsequently evaluated through experimentation.

Basis Beam Selection Methodology

1. Higher topological charges have increased resilience
2. An equal amount of varying orders of the Laguerre polynomial should be selected for pattern diversification
3. Adjacent topological charges should not be selected to avoid similarities in beam design
4. Adding a negative topological charge every 3rd basis beam creates petals in the alphabet patterns, shown to be preferred by the CNN
5. After the alphabet is generated, it should be checked with a correlation matrix to find the lowest average intensity pattern correlation

Section 8: Improving Data Transfer Rate

Section 8A: 64-beam Alphabet

Initially, we had planned to start by evaluating a 64-beam alphabet, as a product of the difficult nature of collecting data prior to synchronization of the camera and SLM. As discussed in Sect. 4A, synchronization of the camera and SLM allowed us to jump straight to collection and processing of 256-beam alphabets. Preliminary testing with 64-beam alphabets indicated we would be highly successful, as there is very little change from previous work. Therefore, this report will focus on evaluating our basis beam selection methodology and improving the data rate by using 256 and 1024-beam alphabets.

Section 8B: Design of 256-beam Alphabets

Our basis beam selection methodology was put into practice by designing three distinct alphabets. The first, Alphabet A, is expected to perform the best, and was selected using the above-mentioned methodology. The resulting basis beams are $\{U_{0,1} U_{1,9} U_{2,-7} U_{3,9} U_{0,-8} U_{1,1} U_{2,5} U_{3,6}\}$, shown in Fig. 8.1A. The second, Alphabet B, is expected to perform moderately. For this alphabet, the orders of the Laguerre polynomials were diversified, but low topological charges were chosen to limit resilience and diversification. The selected basis beams are $\{U_{0,1} U_{1,5} U_{2,-3} U_{3,5} U_{0,-4} U_{1,1} U_{2,2} U_{3,3}\}$, shown in Fig. 8.1B. Finally, Alphabet C is expected to perform the worst and was selected using a small diversity of orders of Laguerre polynomials, and adjacent topological charges. The selected basis beams for Alphabet C are $\{U_{0,1} U_{0,2} U_{0,-3} U_{0,4} U_{1,1} U_{1,-2} U_{1,3} U_{1,4}\}$, shown in Fig. 8.1C.

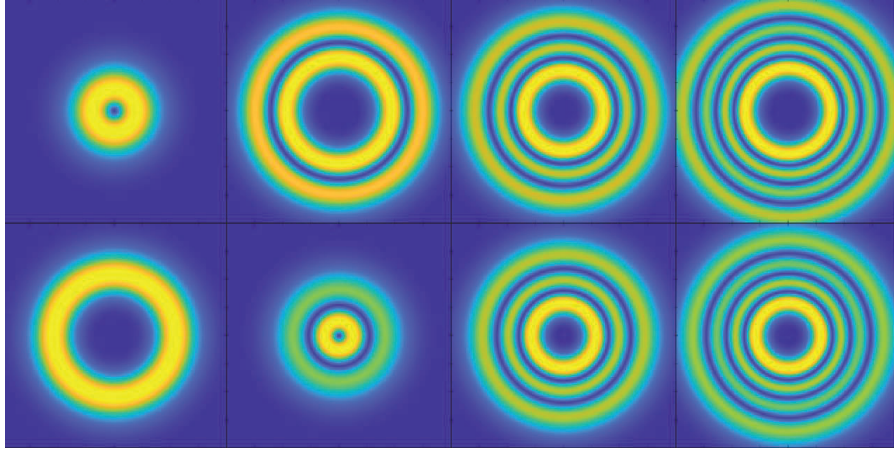


Figure 8.1A: Theoretical images of basis beams for Alphabet A

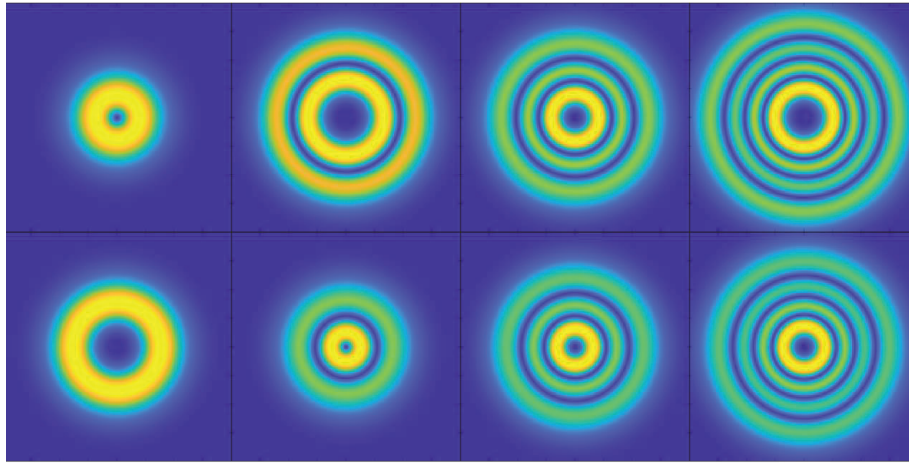


Figure 8.1B: Theoretical images of basis beams for Alphabet B

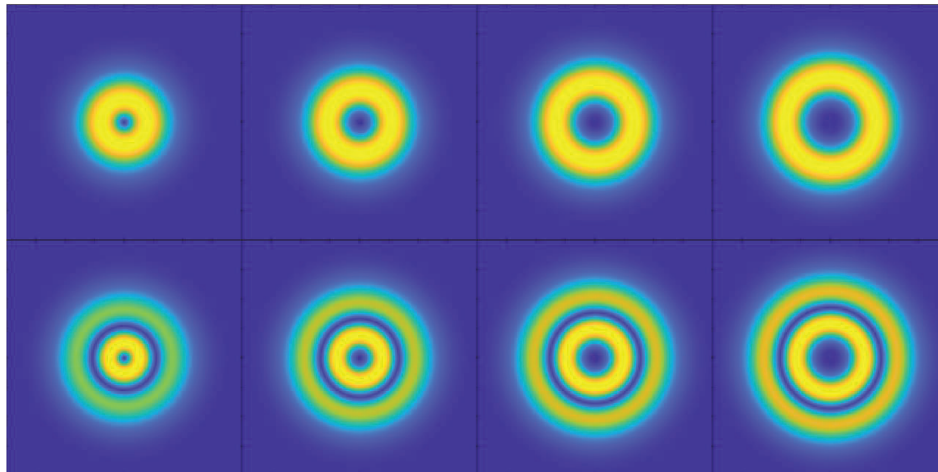


Figure 8.1C: Theoretical images of basis beams for Alphabet C

Given the CNN's use of the intensity profile of the images, it is hypothesized that alphabets with a lower intensity profile correlation will perform better. The correlation matrices for alphabets A, B, and C, given in figures 8.2A, 8.2B, and 8.2C, provide preliminary validation for our basis beam

selection methodology. Alphabet A, the “best” alphabet, has the lowest average correlation of 0.71. Alphabet B has an average correlation of 0.77, and Alphabet C has an average correlation of 0.81. Each alphabet has a higher correlation (shown in yellow) at the lower end, as these beams are a result of six, seven, or eight basis beams superimposed, creating less variation.

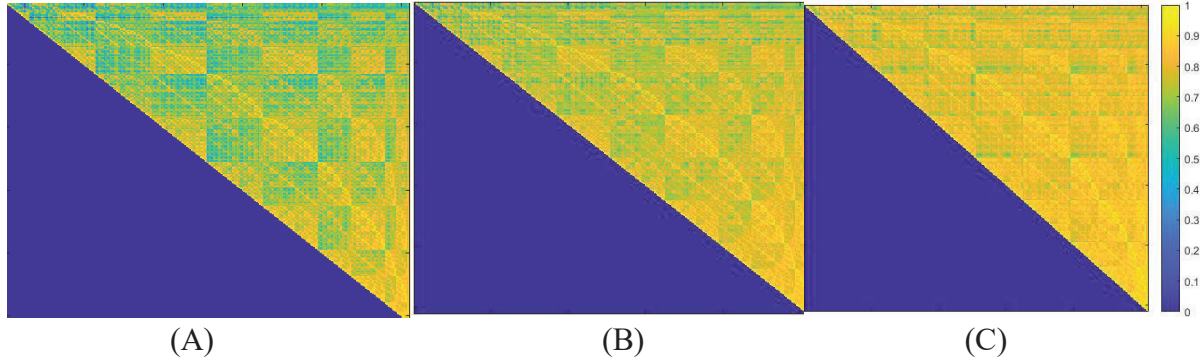


Figure 8.2: Correlation matrices for Alphabet A (A), Alphabet B (B), and Alphabet C (C). Correlating beams goes from 1 to 255, left to right and top to bottom. Yellow indicates a higher correlation, blue and green indicate lower correlation

For comparison, the full alphabets for Alphabets A, B, and C are given in Fig. 8.3, Fig. 8.4, and Fig. 8.5 respectively. As will be described in further detail later, these alphabets will be evaluated through similar conditions, and their performance will be quantified and analyzed using the same CNN, retrained in each scenario.

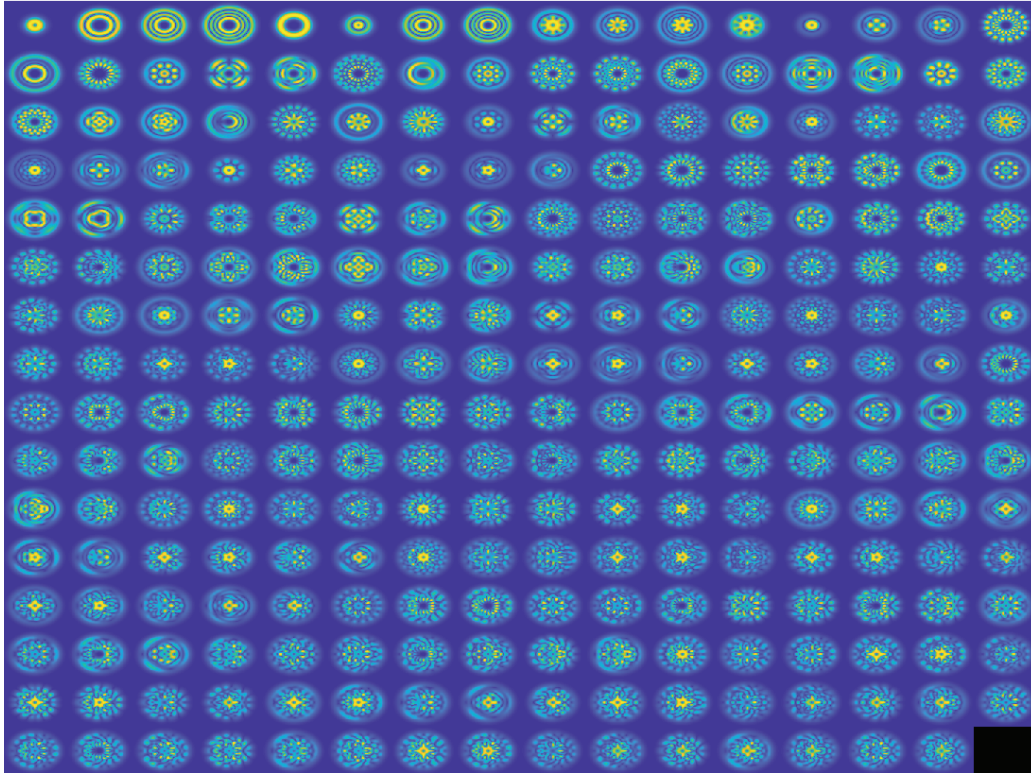


Figure 8.3: Theoretical images of Alphabet A

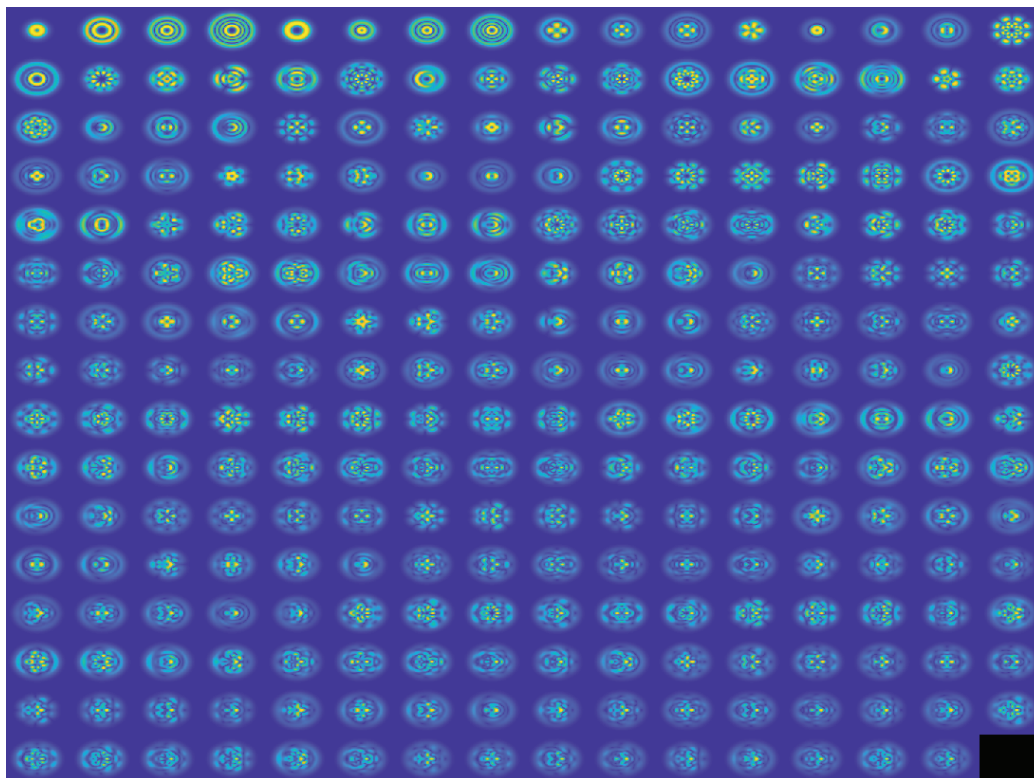


Figure 8.4: Theoretical images of Alphabet B

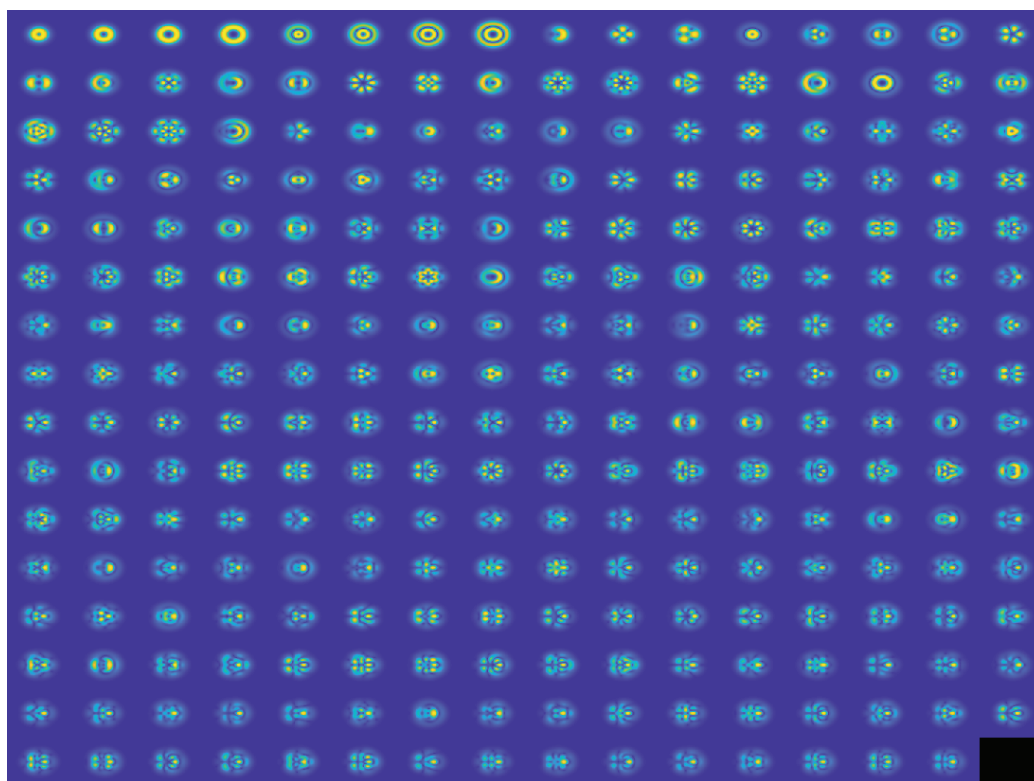


Figure 8.5: Theoretical images of Alphabet C

Section 8C: Evaluation of 256-beam Alphabets through Simulation

First, Alphabet A was evaluated through simulation, with parameters $N = 1024$, $D = 10 \text{ cm}$, $Z = 30 \text{ m}$, $n_{\text{scrns}} = 10$, and $\lambda = 633 \text{ nm}$, $\varepsilon = 1 \times 10^{-5} \left[\frac{\text{m}^2}{\text{s}^3} \right]$ and $\chi_T = 1 \times 10^{-7} \left[\frac{\text{K}^2}{\text{s}} \right]$. 500 images of each beam were collected, and a CNN was trained on 100 images of each, with the remaining images used for testing. The CNN used was the same as Sect. 6C, Fig. 6.8, with the same training parameters. The CNN was able to quickly learn and classify the 256-beam alphabet, with no trouble. This is highlighted by the speed at which the training graph reaches 100%, shown in Fig. 8.6, meaning the variation between images within a beam were small, and the beams themselves were distinct. Following the ease of classification via simulation, and the significant computational resources and time required to generate 128,000 distinctly impacted images through simulation, the decision was made to move to experimental work, rather than running Alphabets B and C through simulation, as we expected to receive the same inconclusive result.

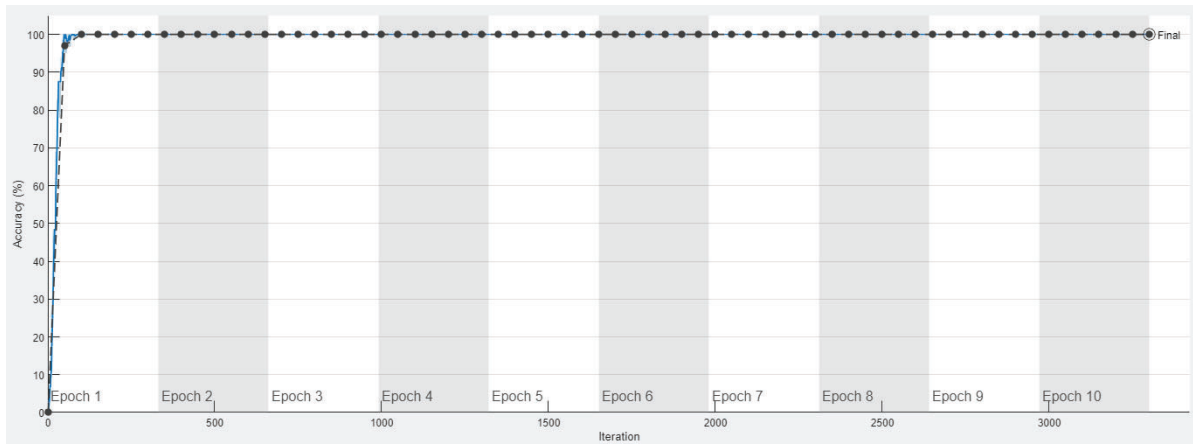


Figure 8.6: 256-beam alphabet training graph for Alphabet A

Section 8D: Experimental Evaluation of 256-beam Alphabets and Turbulence Validation

The three alphabets were propagated through our experimental set-up, and subjected to similar turbulence. For each experiment, the data was collected by cycling through the 256 beams in each alphabet, with beam zero corresponding to a Gaussian beam, used for estimating the scintillation. The beams were cycled on the SLM at a rate of 20 Hz, meaning the beam pattern was changed every 50 ms. Within each cycle, 11 images of said beam were collected by the camera, totaling 40 ms for each collection. The extra 10 ms were added to the end in order to minimize capture error by the camera. Total collection time for each dataset was 21.25 minutes, with 281,600 total images collected, the result of 100 full loops through the alphabet. However, when the data was processed, the 11th frame of each collection was cut out to minimize synchronization error and crossover between adjacent beams as the SLM switched, providing 256,000 usable images, or 1000 per alphabet symbol. Data download time between experiments was approximately 2 hours, which allowed the tank temperature to normalize between experiments. Figures 8.7, 8.8, and 8.9 provide examples images of Beam 35 from alphabets A, B, and C, highlighting the variation between the alphabets and one instance of how the turbulence distorted the beams.

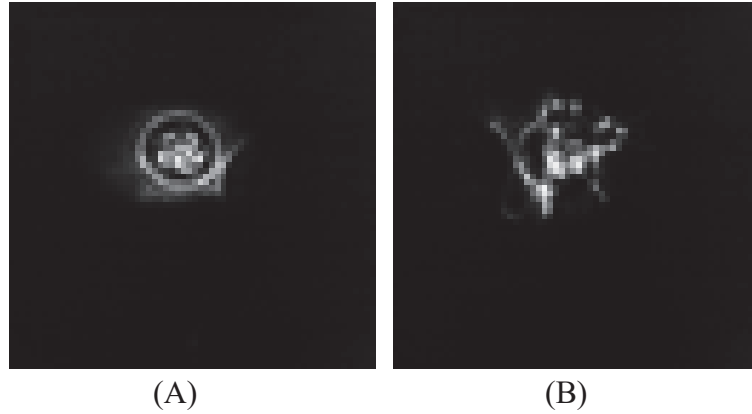


Figure 8.7: Beam 35 from Alphabet A in (A) weak induced turbulence and (B) strong induced turbulence

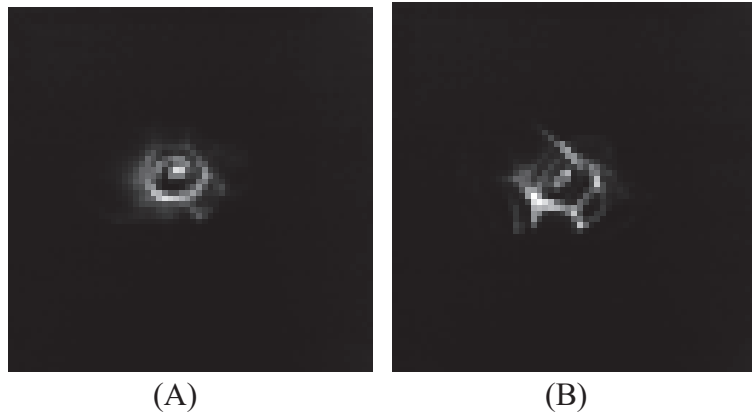


Figure 8.8: Beam 35 from Alphabet B in (A) weak induced turbulence and (B) strong induced turbulence

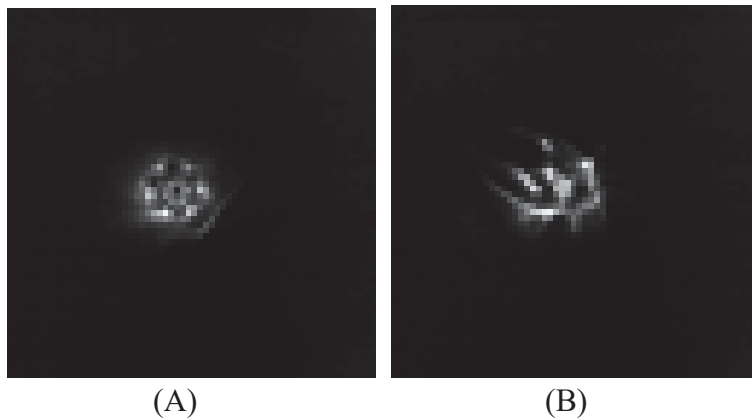


Figure 8.9: Beam 35 from Alphabet C in (A) weak induced turbulence and (B) strong induced turbulence

The optical turbulence in this experiment is generated using the water heater placed in the tank. For each of the experiments, the heater was set to 140° F, with the starting water temperatures summarized in Table 8.1.

Alphabet	Initial Water Temperature (°F)
A	71.9
B	72.4
C	71.7

Table 8.1: Summary of starting water temperatures

To further evaluate the consistency of the experiments, the scintillation index (SI) of a Gaussian beam was calculated and used to find the approximate refractive-index structure constant C_n^2 for each experiment, following the methodology outlined in Sect. 3C. The time-averaged and normalized Gaussian beam for each experiment is provided in Fig. 8.10.

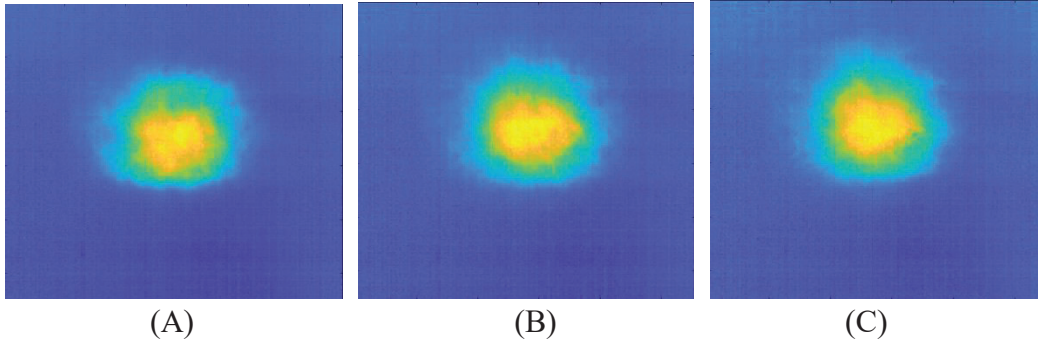


Figure 8.10: Time-averaged and normalized mean intensities of Gaussian beam from experiments A, B, and C

The normalized mean intensities of the Gaussian beam from experiments A, B, and C are shown in Fig. 8.10A, B, and C respectively. These images represent a time average of each pixel of the 1000 frames collected of a Gaussian beam for each of the three experiments. As can be seen, there is a strong similarity between the three images, indicating similarity between the three trials, and providing validity for the approximation of the center of the Gaussian beam as a flat surface used in the SI calculation.

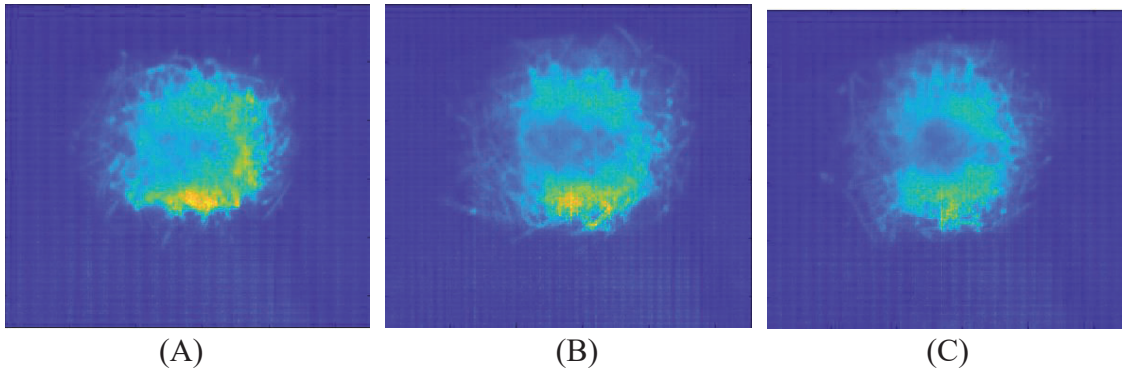


Figure 8.11: Time-averaged SI of Gaussian beam for experiments A, B, and C

Fig. 8.11A, B, and C represent a time-average calculation of the SI of the 1000 frames collected of a Gaussian beam for each of the three experiments. Numerically, the peak SI value is across

the three trials is approximately the same, and the shape of the scintillation follows the same pattern as well. The heater generates optical turbulence approximately 5 cm below the propagating beam, which then rises. This is highlighted in the stronger vertical scintillation of the Gaussian beam in each experiment, though there still is significant horizontal oscillation. The time and spatially-averaged estimation for the refractive index structure constant is summarized in Table 8.2.

Alphabet	C_n^2 Value ($m^{-2/3}$)
A	6.96×10^{-11}
B	6.78×10^{-11}
C	6.83×10^{-11}

Table 8.2: Time (full collection) and spatially (center 21 x 21 pixels) averaged C_n^2 values for each experiment

As can clearly be seen, the averaged C_n^2 values are similar, indicating similar levels of turbulence through the experiments. These numbers do not corroborate the measurements taken in Ref. [2], which used thermocouples to record the temperature fluctuations created by the same heater, and estimate the refractive-index structure constant. The authors in this study found the value to range from 1×10^{-7} to $1 \times 10^{-8} m^{-2/3}$. However, this error is likely the result of errors in both estimation and measurement.

In addition to time-averaging across the full experimental timescale, the SI was time-averaged in 10 frame chunks, or every 12.8 seconds, for the full collection time. This was done to provide a quantitative analysis of how the turbulence changed over time in each experiment. As can be seen in Fig. 8.12, the turbulence varies randomly, but stays within the same order of magnitude, as would be expected. Between the three experiments the C_n^2 values also fluctuate to the same degree of magnitude, indicating that while the turbulent fluctuations between trials are different, they are of similar strength and size, therefore impacting the propagated beams in a similar fashion. The high levels of accuracy in strong turbulent fluctuations provides evidence that the system would perform better, not worse, in practical conditions.

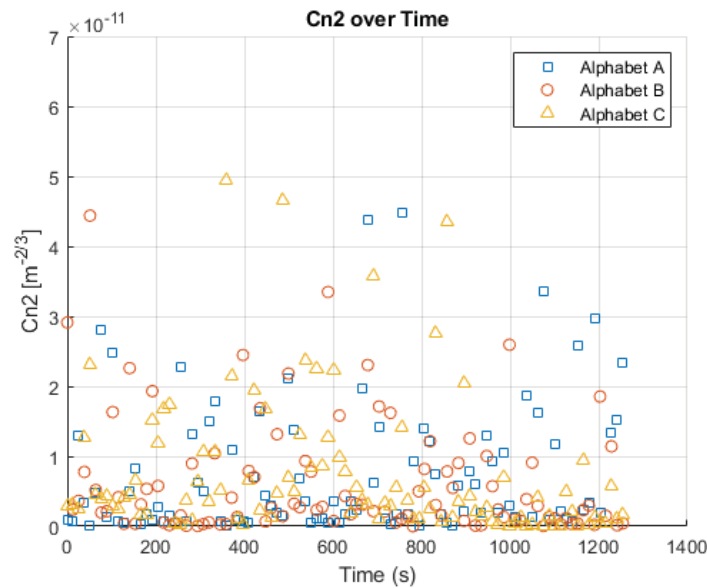


Figure 8.12: C_n^2 fluctuations over time in 10 frame chunks(12.8 seconds)

Section 8E: 256-beam Alphabet Results and Conclusions

The goal of this study was to experimentally determine whether the selection of our basis beams has a quantifiable effect on performance when the beams are propagated through similar optical turbulence. Additionally, we record the trade-off between time and classification accuracy as the number of total training images increases. For each scenario, our data set was randomly split into a testing and training dataset. The training dataset contained the listed number of images of each beam, while the testing dataset was selected from the remainder of the images. From the testing dataset, a validation dataset was drawn consisting of 10% of the total training images. From the testing data set, 400 images of each beam were randomly drawn for testing in order to create a testing dataset of approximately 100,000 images. For each of the following training / testing scenarios, our network was retrained from scratch prior to testing, in order to maintain consistency within trials. The results are contained below in Table 8.3.

Alphabet	Size of Training Dataset (imgs)	Training Time (min:sec)	Size of Testing Dataset	Classification Accuracy
A	76,500 (300 / beam)	11:46	102,000	92.12%
B	76,500	12:56	102,000	91.79%
C	76,500	13:26	102,000	89.74%
A	127,500 (500 / beam)	20:00	102,000	96.92%
B	127,500	19:42	102,000	97.09 %
C	127,500	21:16	102,000	96.16%
A	178,500 (700 / beam)	23:45	76,500	98.32%
B	178,500	37:41	76,500	98.21%
C	178,500	41:16	76,500	97.26%

Table 8.3: Summarized CNN Results

As shown, there is little distinction between Alphabets A and B, the “best” and “moderate” alphabets. This indicates that diversification of the alphabet symbols, rather than selection of higher topological charges has a bigger impact on the alphabet’s performance. Both alphabets performed noticeably better than Alphabet C, which validates our diversification theory. Additionally, this lends evidence to using correlation matrices as a preliminary predictor of performance, but it clearly is not the only determining factor. This is because the correlation is completed on static, undistorted images, and in practice the “lines” between beams can be blurred. Figures 8.13, 8.14, and 8.15 provide inaccuracy graphs for each scenario. Like with the basis beams, there is no clear trends regarding which beams were the most misclassified. This is in opposition to what we hypothesized, which was that beams at the higher end of the alphabet would be misclassified more often, as they are correlated higher with the beams around them as a product of many basis beams.

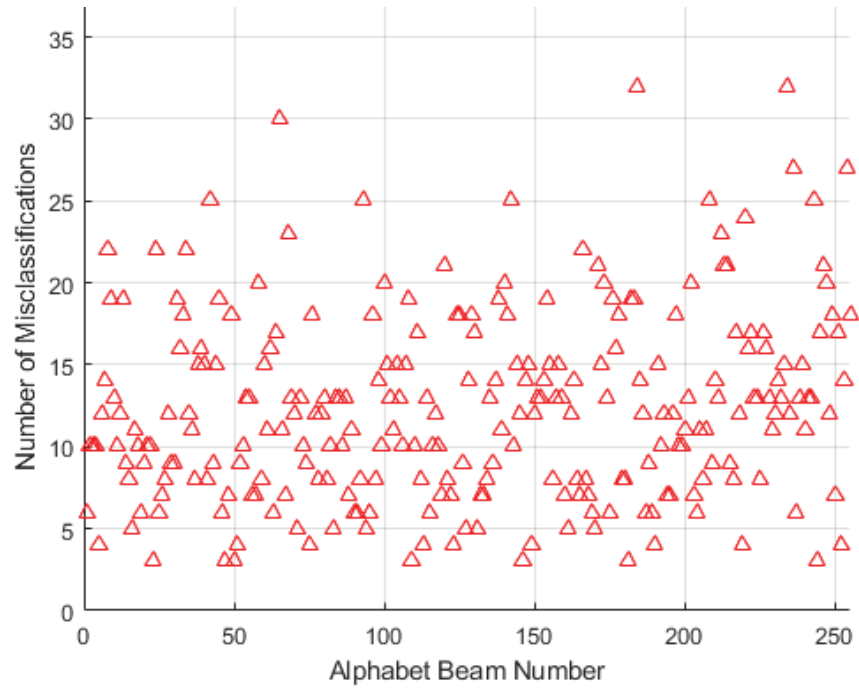


Figure 8.13: Inaccuracy graph for Alphabet A representing number of misclassifications per beam in alphabet

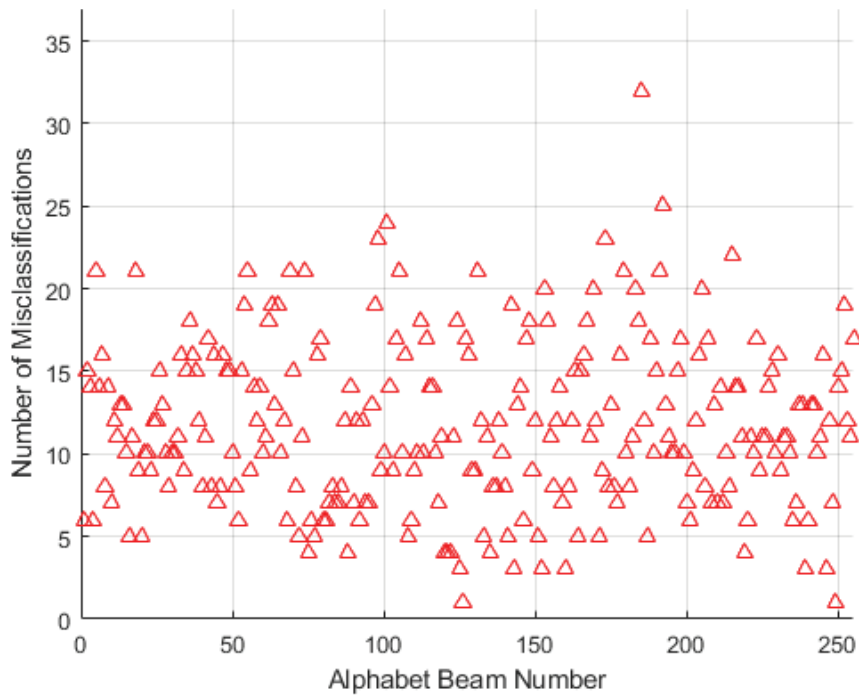


Figure 8.14: Inaccuracy graph for Alphabet B representing number of misclassifications per beam in alphabet

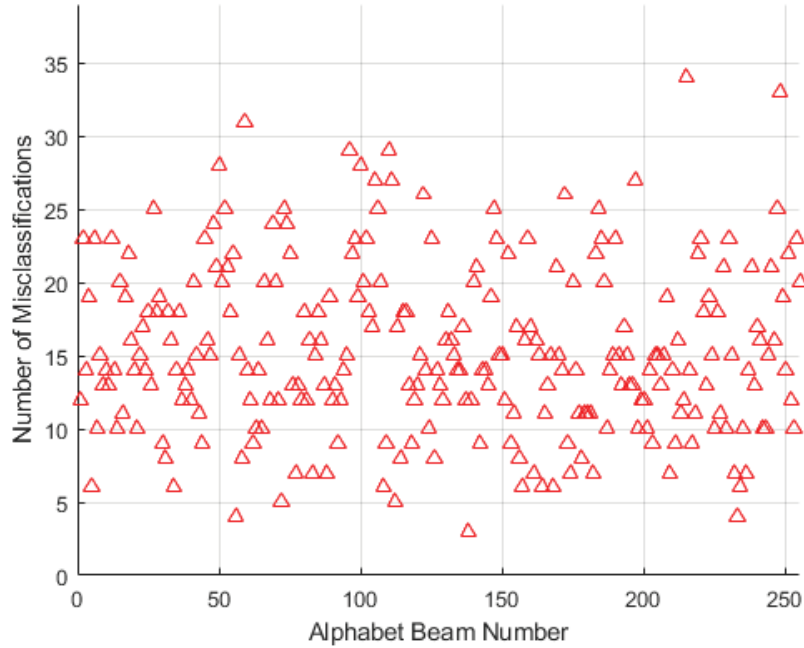


Figure 8.15: Inaccuracy graph for Alphabet C representing number of misclassifications per beam in alphabet

Section 8F: 1024-beam Alphabet Evaluation and Performance

Having propagated 256-beam alphabets through significant optical turbulence with a high level of success, we turned our attention to designing and propagating a 1024-beam alphabet. The alphabet was designed using our basis beam selection methodology, resulting in the following 10 basis beams: $\{U_{0,1} U_{1,9} U_{2,-7} U_{3,9} U_{0,-8} U_{1,1} U_{2,5} U_{3,6} U_{0,4} U_{1,-5}\}$, shown in Fig. 8.16. The full alphabet is provided in Fig. 8.17 for a visual comparison to the alphabets given in figures 8.4-8.6.

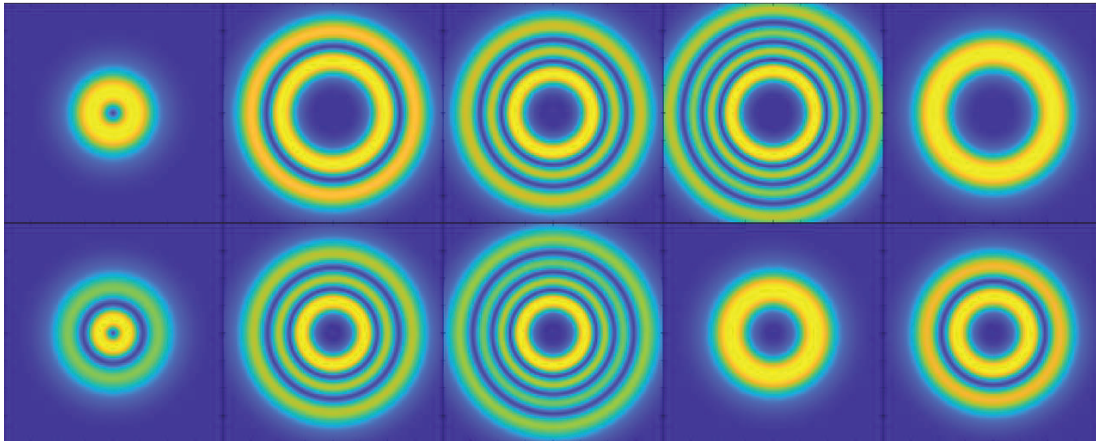


Figure 8.16: Theoretical images of basis beams for 1024-beam alphabet

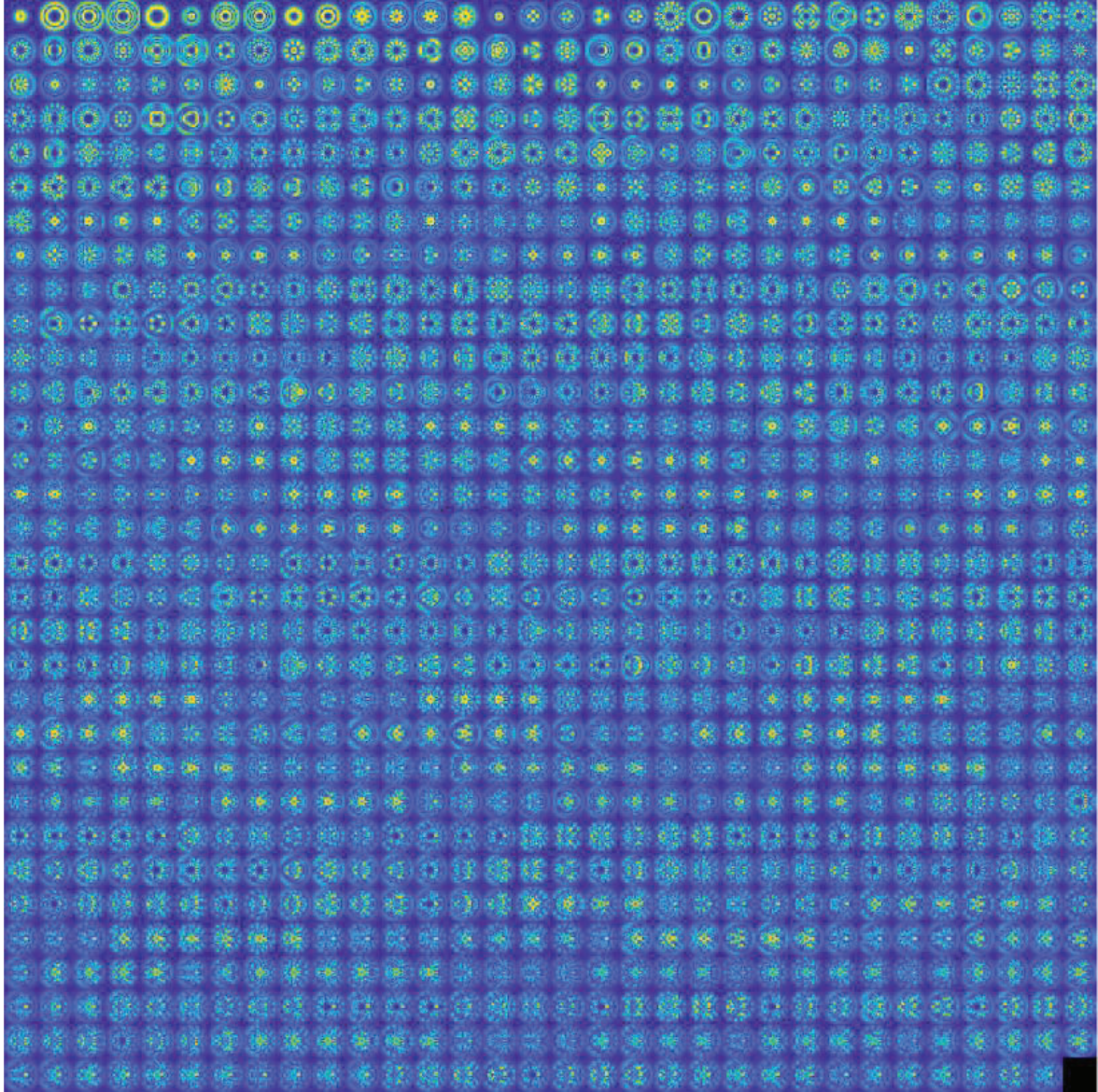


Figure 8.17: 1024-beam alphabet

Once generated, the alphabet was checked using a correlation matrix to ensure a relatively low correlation. While the average correlation (0.73) is higher than Alphabet A (0.71), it is still relatively low, given the large number of basis beams and total symbols. This correlation matrix is provided in Fig. 8.18.

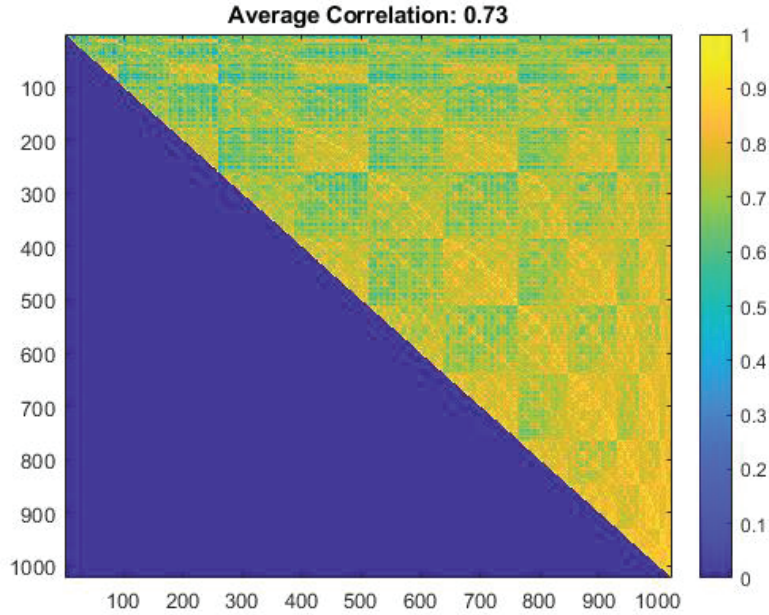


Figure 8.18: Correlation matrix for 1024-beam alphabet with an average correlation of 0.73

The 1024-beam alphabet was evaluated using our experimental set-up. The beams were cycled on the SLM at a rate of 20 Hz, meaning the beam pattern was changed every 50 ms. Within each cycle, 11 images of said beam were collected by the camera, totaling 40 ms for each collection. The extra 10 ms were added to the end in order to minimize capture error by the camera. The dataset was collected in two chunks, as the SLM was only capable of holding half the screens at once (513). This was likely due to an issue with RAM in the computer running the SLM. A Gaussian beam was added to the start of each loop to estimate turbulence through the span of the collection time, as well as to compare the first half to the second. Total collection time for each dataset was 42.75 minutes, with 564,300 total images collected per half, the result of 100 full loops through the alphabet. The total collection time was 85.5 minutes, with 1,127,500 total images collected. When the data was processed, the 11th frame of each collection was cut out to minimize synchronization error and crossover between adjacent beams as the SLM switched, providing 1,025,000 usable images, or 1000 per alphabet symbol. Data download time between experiments was approximately 6 hours. The CNN was trained using half the images for training and half for testing, and resulted in an impressive classification accuracy of 91.26%, summarized in Table 8.4. The confusion matrix for the 1024-beam alphabet was unreadable due to the size of the matrix, but the inaccuracy graph, Fig. 8.19, provides insight into the misclassifications. First, it is clear the turbulence impacted the beams significantly more during the second collection of data, as there is a stark jump in misclassifications at beam 513. Additionally, most of the misclassifications are concentrated between 0 and 50, indicating no major trends. Beam 633 had the greatest number of misclassifications, at 357, but there is nothing visually about this beam that would indicate why this outlier occurred. In order to determine this, repeated testing under better controlled experimental conditions should be completed.

Alphabet	Size of Training Dataset (imgs)	Training Time (min:sec)	Size of Testing Dataset	Classification Accuracy
1024	511,500 (500 / beam)	131:04	511,500	91.26%

Table 8.4: Summary of CNN results for 1024-beam alphabet

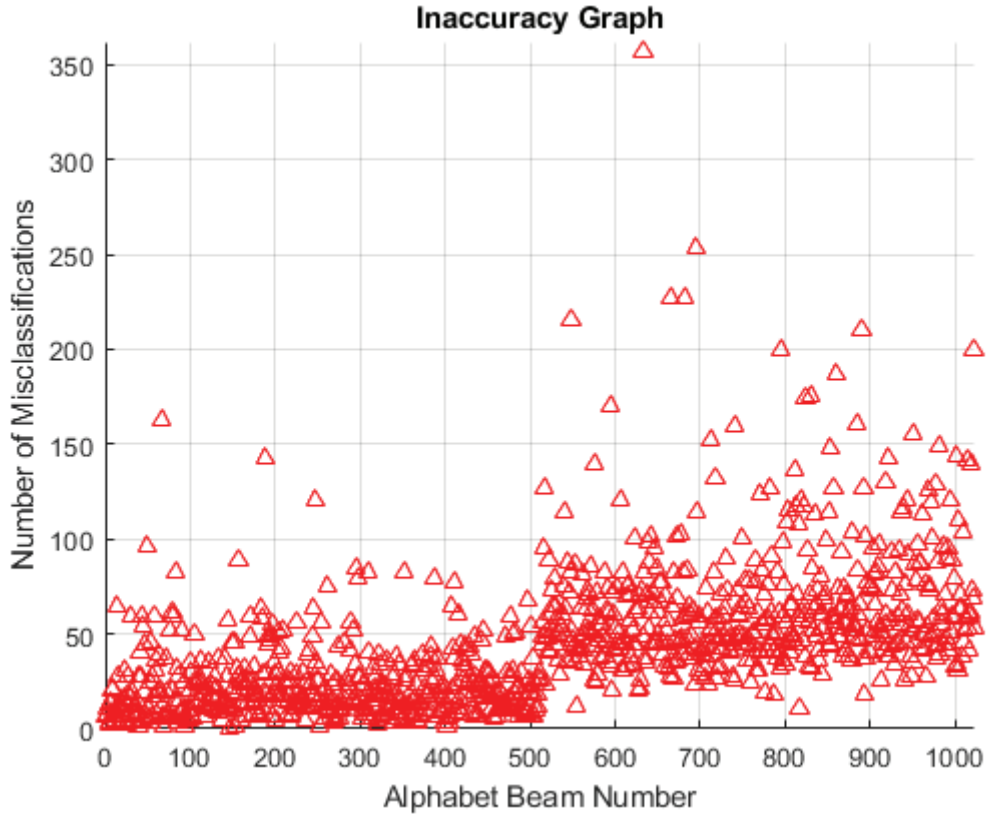


Figure 8.19: Inaccuracy graph for 1024-beam CNN testing

In order to validate our results and compare the separated data collections, scintillation of a Gaussian beam placed at the start of every loop was collected. As discussed previously, this was used to estimate the magnitude of the refractive index structure constant. This is provided in Table 8.5. As shown, the turbulence is estimated to have been stronger on the second half, likely due to the increased water temperature. For future collections, more time should be taken to let the water temperature settle, even if it increases the time between collections. Additionally, these discrepancies highlight the importance of using a computer with enough RAM to store and cycle all the screens on the SLM at once. In Fig. 8.20, the time-averaged and normalized intensities of the Gaussian beam from the first and second half collections. While the beam shape is similar, there appears to be a slight beam misalignment, which may have negatively impacted second half classifications. Beam misalignment is discussed further in the next section. In Fig. 8.21, the pixel-by-pixel scintillation indices for the two collections are compared. Though similar, the SI from the second collection is larger and of a greater peak magnitude. This indicates stronger turbulence in the hotter water, and is likely the primary cause of a larger number of misclassifications.

Alphabet	C_n^2 Value ($m^{-2/3}$)	Starting Water Temp ($^{\circ}\text{F}$)
1 st Half	1.11×10^{-10}	72.0
2 nd Half	3.62×10^{-10}	83.0

Table 8.5: Time and spatially-averaged refractive index structure constant values for the 1st and 2nd collections, related to the starting water temperature

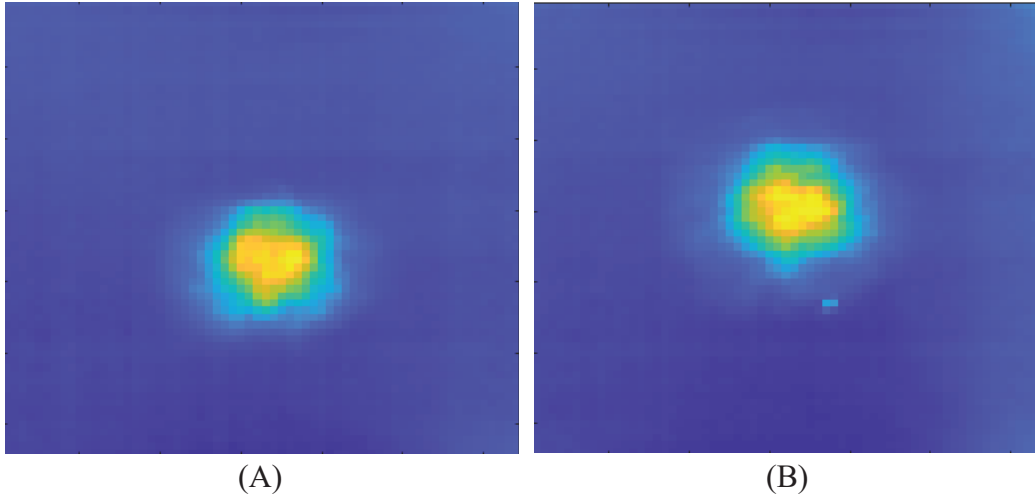


Figure 8.20: Time-averaged and normalized Gaussian beam for (A) first half of collection and (B) second half of collection

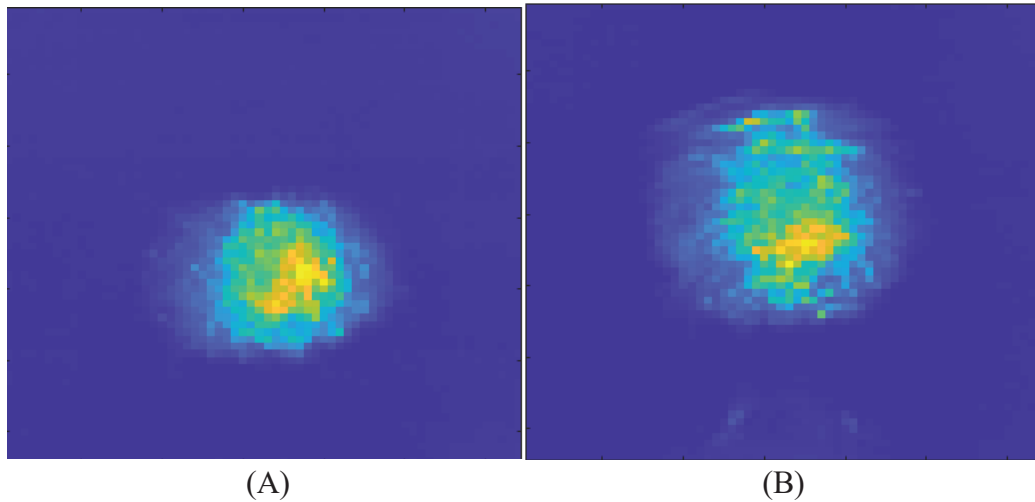


Figure 8.20: Time-averaged pixel-by-pixel scintillation index for (A) first half of collection and (B) second half of collection

The 1024-beam alphabet was able to achieve over 91% accuracy using only 500 images per beam for training. While this accuracy level isn't high enough to be used in a system today or tomorrow, it is promising and indicates the strength and power of CNNs for dealing with complex and large systems. At 1024-beams the variations become so small it would be time consuming and difficult for the human eye to differentiate, but the CNN can accomplish this with only taking in an input of 64 x 64 pixels.

Section 9: Beam Misalignment

When conducting my experiments for the 256-beam alphabets, I began by conducting a preliminary test (Alphabet A) in December 2021. When I returned from winter break in January 2022, I decided to recollect Alphabet A, providing me with two datasets of the same alphabet. However, during the break, the system had become misaligned, in a way that only affected where the beam was located on the screen the camera captured. In other words, the beam was translated slightly (approximately 1.5 cm) along the y-axis. In my experimentation, I decided to train on one collection and test on the other to see how the CNN performed. To my surprise, the CNN performed extremely poorly. For both cases, the CNN would achieve a classification accuracy of $\sim 98\%$ when tested on its own dataset, but $\sim 1\%$ when tested on the other dataset, which is better than randomly guessing, but not by much. We were unable to find any research where the authors utilize a testing set that was not collected at the same time as the training set, indicating an area for important research. For a system to be practically used, it must be able to handle translations of the signal, as long as they remain within the parameters of the receiver. This led us to design an experiment and investigate beam misalignment, the effect on performance, and strategies for improving classification accuracy.

Section 9A: Design and Set-Up of Experiment

In order to test beam misalignment quickly but effectively, we opted to use a 16-beam alphabet, shown below in Fig. 9.1. This allowed for smaller and faster datasets, and a simplification of the classification problem.

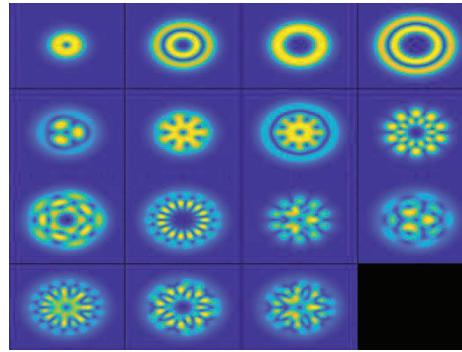


Figure 9.1: 16-beam alphabet used for beam misalignment experiment

To evaluate the effect of beam misalignment, we collected three datasets in quiescent water. Fig. 9.2 shows an example of Beam 5 from each dataset. The first dataset was centered in the camera frame, providing our baseline dataset, Dataset Center. The second was translated approximately 3 centimeters on the screen in the positive x-direction, creating Dataset Right and the third was translated approximately 3 centimeters in the negative y-direction, creating Dataset Down. For each collection, 1000 images of each beam were collected and processed for the CNN.

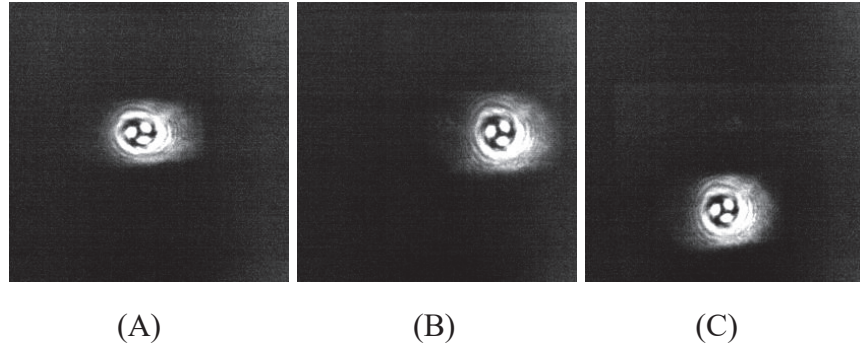


Figure 9.2: (A) Beam 5 centered on camera Dataset Center, (B) Beam 5 translated in the positive x-direction for Dataset Right, and (C) Beam 5 translated in the negative y-direction for Dataset Down

Section 9B: CNN Performance

In tracking performance, the CNN was trained in three different scenarios, using only data from Dataset Center. For the first training, the CNN was provided images from Dataset Center, which were not augmented in any way. The CNN was then tested on 10,050 images from each dataset and the classification accuracies recorded in Table 9.1. Following this, the CNN was trained using augmented data, which randomly translated the training data in the x and/or y-direction ranging 20 pixels in either direction $[-20\ 20]$. This created a varied and translated data set. When tested on the center data, the CNN performed slightly worse, but when tested on the Right and Down datasets, the CNN performed much better. This is because it had seen data that was similar to the testing material, even though it had never seen data from either the Right or Down datasets. An example of augmentation in the training set is shown in Fig. 9.3. Finally, the CNN was trained using “smart” augmentation, meaning the training dataset was only augmented to move $[0\ 20]$ pixels in the +x-direction or -y-direction. This was to maximize the amount of similar data to what I knew would be expected for the testing. Interestingly, the CNN actually performed worse, as recorded in Table 9.1. Additionally, training time increased with augmentation indicating increased complexity for the CNN.

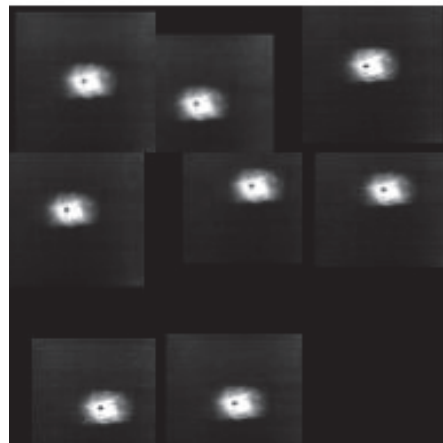


Figure 9.3: Example of augmentation of Dataset Center to improve performance

Training Set (300 Images / Beam, 4500 Images Total)	Training Time (min : sec)	Testing Set (10,050 Images) Classification Accuracy		
		Center	Right	Down
No Augmentation	0:35	100%	6.46%	4.64%
Augmentation	2:34	99.35%	89.29%	72.73%
“Smart” Augmentation	2:48	99.67%	86.31%	61.35%

Table 9.1: CNN performance for various training scenarios

Figures 9.4A and 9.4B are confusion matrices for the “augmentation” testing set, indicating that the CNN misclassified when it had learned one beam to be another. This likely shows that the CNN needed more augmented training images because the images for that beam were not randomly augmented in a way that allowed the CNN to learn the translation.

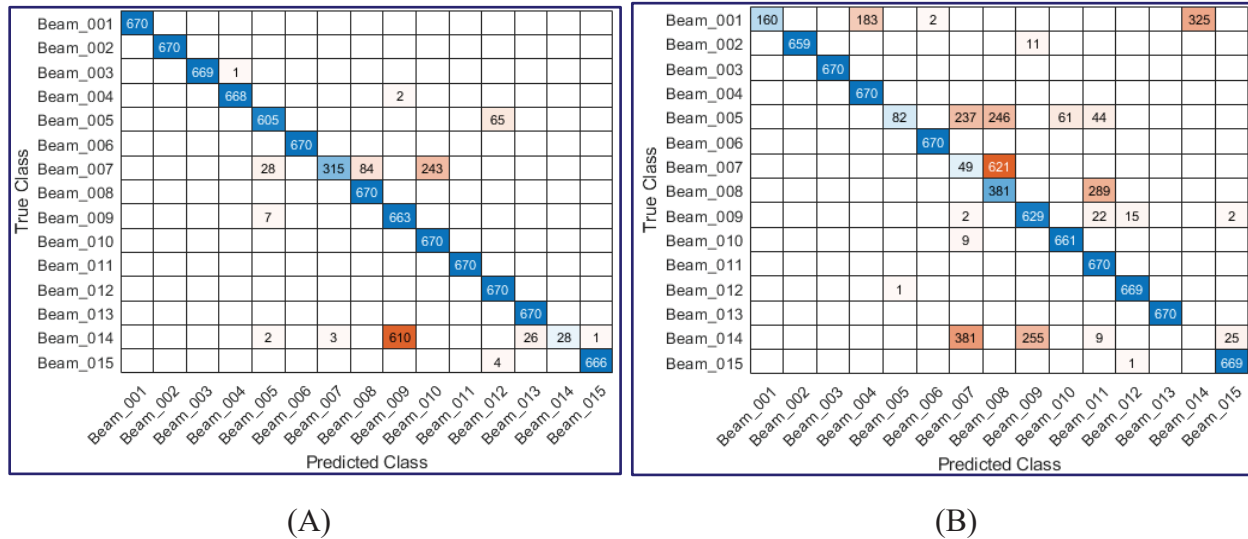


Figure 9.4: (A) Confusion matrix for augmented training set and testing on Dataset Right and (B) confusion matrix for augmented training set and testing on Dataset Down

Section 9C: Conclusions and Next Steps

From this work we concluded that while great care must be taken in the training and testing of your CNN, it is possible to train a CNN that will be robust and resilient to changes to the environment. The key factor in doing this is to expose your CNN to as many likely changes to the environment as possible, as it struggles to learn what it hasn’t seen. As shown all throughout Section 8, CNNs are capable of learning and “memorizing” all manners of variations, even when

turbulence is significant or misalignment occurs during the collection process. The larger the dataset you are able to produce for your CNN, the more accurately it will be able to classify, especially on data the CNN has never seen before. The next step along this project would be to repeat the same experiment but under turbulent conditions, rather than in quiescent water. Additionally, the beam misalignment should be completed using the 256-beam alphabet and a much larger, augmented or partially augmented dataset, on the order of 5000 training images per beam. This would likely allow you to refine the beam's ability to learn all the alphabet's patterns, as well capturing the full scope of turbulent variations.

Section 10: Construction of a Rayleigh-Bénard Turbulence Tank at Naval Research Laboratory, Washington, D.C.

One of the greatest challenges in evaluating an underwater wireless optical communication system is subjecting the system to realistic turbulence. As discussed, many researchers use simulated turbulence which aims to stochastically capture the variations observed in the true environment. This has its drawbacks, as it assumes the flow is homogenous and isotropic, which is not necessarily the case, so it is only valid for weak fluctuations. In order to strengthen the turbulence, one must propagate over unrealistic distances, as most simulations do not account for attenuation. In the experimental case, it is difficult to recreate oceanic turbulence. We make use of a heater which produces strong, localized turbulence. While this is good for providing strong distortions to our beams, it is not necessarily the turbulence the beam would experience propagating over longer distances through the ocean. This has led us to pursue building a Rayleigh-Bénard (RB) turbulence tank, which produces turbulent flow from natural convection, providing a more accurate estimate for realistic distortions on our communication system.

Section 10A: Rayleigh-Bénard Turbulence

RB turbulence is the product of a hot layer on the bottom, causing fluid to rise, and a cold layer on the top causing the fluid to fall. This produces large turbulent cycles that distort the light as it propagates through, shown in Fig. 10.1.

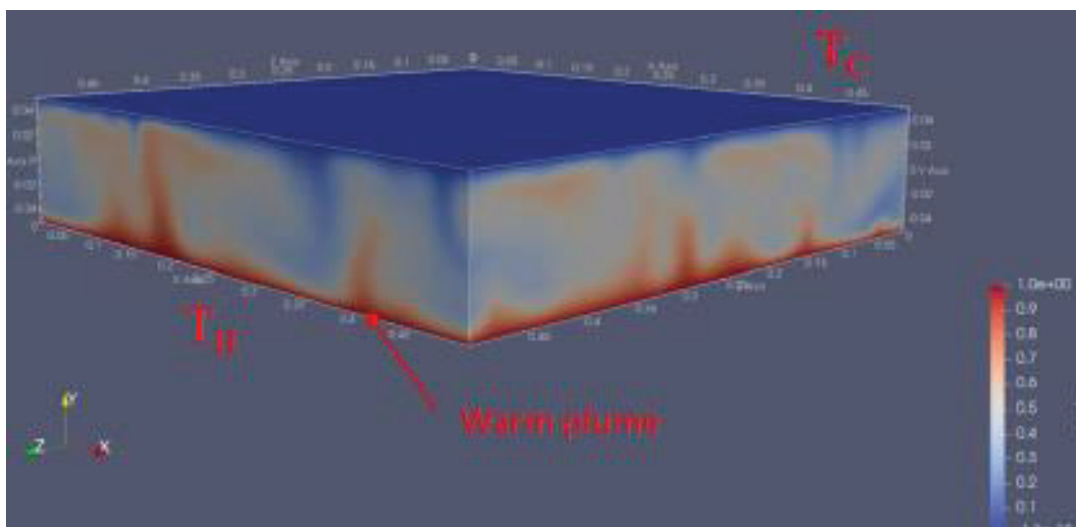


Figure 10.1: Simulated Rayleigh-Bénard Convection [48]

Researchers at NRL, Washington D.C., NRL, Stennis Space Center, George Mason University, and the United States Naval Academy have collaborated to produce an RB simulation. The researchers utilized several methods of simulating the turbulence, namely using OpenFoam to solve the general system of equations that govern the dynamics of the turbulence. Fig. 10.2 provides cross sections from the simulated turbulence inside of a tank. From these cross-sections, phase screens can be drawn along where the beam would propagate, to create an RB split-step simulation. An example of these phase screens is given below in Fig. 10.3.

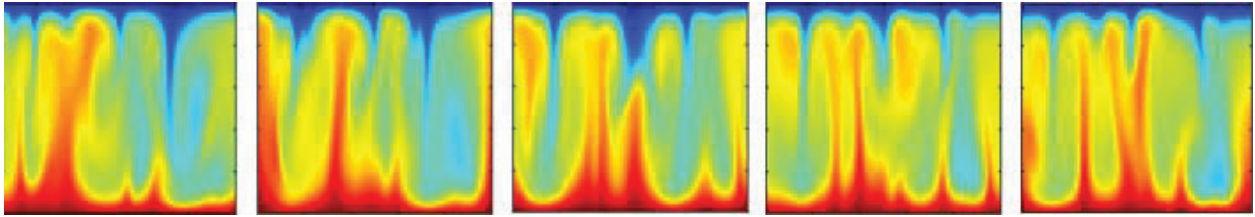


Figure 10.2: RB turbulence cross sectional representations

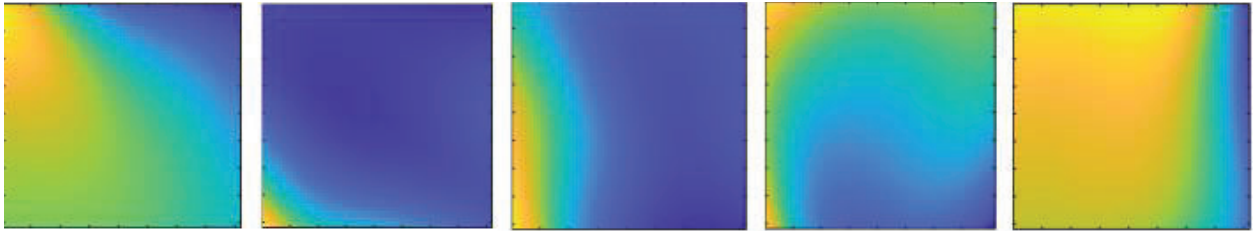


Figure 10.3: Phase screens drawn from RB cross-sections

These simulations are the focus of the manuscript currently in preparation by Dr. Avramov-Zamurovic (USNA), Dr. Judd (NRL), Dr. Matt (Stennis), Dr. Handler (GMU), Dr. Watnik (NRL), Dr. Lindle (NRL), Dr. Esposito (USNA), and myself.

Section 10B: Tank Construction

My primary role in the RB turbulence project has been to help construct an RB turbulence tank at NRL. I have been able to travel to NRL biweekly over the course of the 2022 academic year, and have almost finished with the construction process. The tank requires several systems to be put together to properly generate turbulence. The tank itself is composed of a top and bottom plate which provide the cool and warm layers that cause turbulence in the water. The top plate will be connected to a water cooler that is temperature controlled (Fig. 10.4), and the bottom plate will be connected to a hot water heater (Fig. 10.5A). Though the tank is not finished yet, all the pieces are acquired and ready for final installation. This includes connecting the water sources to the plates, seeing the tank insulation, and setting up the optical equipment to validate the turbulent flow inside the tank. The tank was designed with longevity in mind, leading to a completed lifting rig to raise and lower the top plate, so the tank can be cleaned or adjusted as one sees fit. The tank in its current form is provided in Fig. 10.4. During my research assistantship with Dr. Avramov-Zamurovic in summer of 2022, I hope to complete work on the tank and perform further experimentation resulting in a manuscript and publication.



Figure 10.4: Cold water chiller for cool RB plate



(A)



(B)

Figure 10.5 (A) Hot water heater for warm RB plate and (B) hot water pump

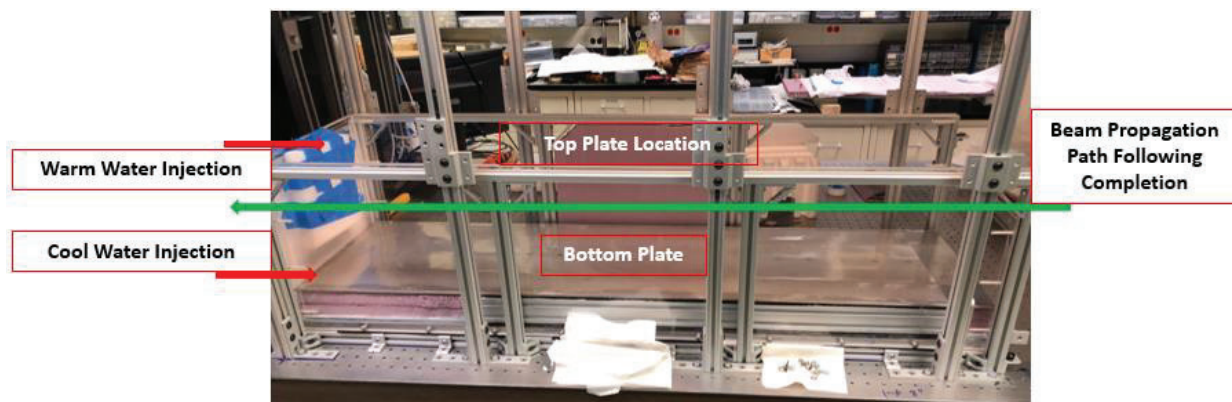


Figure 10.6: Construction of RB turbulence tank

Section 10C: Future Experiments

The work I have done as the bulk of my Trident Scholarship has prepared me to compare the performance of alphabets through atypical turbulence, our heater, to characterized turbulence representative of more realistic conditions. It will also provide ground for comparing results to the work done using the Nikishov spit-step simulation. The first project in the RB tank would be to evaluate the basis beam candidates, using the CNN like before, and also the changes to the beam vortex to determine resiliency trends as it relates to topological charge and order of the Laguerre polynomial. By investigating the effect of the characterized turbulence on the vortex of the beams, we will be able to learn far more about the physics of the light interaction with the turbulence. Understanding this will deepen our understanding of the impacts if different flows, as well as providing a starting place for open or closed loop adaptive optics. Ultimately, a system would be far more effective if it could be adapted to minimize the effect of turbulence on the propagating beams.

Section 11: Conclusions and Future Work

Section 11A: Conclusions

The goal of this project was to explore the design of structured light to improve data transfer rate in underwater wireless communication systems. This was done by increasing the size of a LG beam-based alphabet, optimizing the alphabet, and evaluating performance. The following is an outline of the major project conclusions and novel contributions to the field of underwater wireless optical communication systems.

1. From Sect. 6, we found the CNN moderately preferred the LG beams carrying OAM to a visually similar form of structured light with no OAM. This was likely the result of the differences in how the phase and resulting intensity of the beams was impacted by the induced optical turbulence.
2. From Sect. 7, we found that selection of the basis beams matters for optimal classification accuracies. Following the outlined methodology helps to create robust, diverse, and resilient alphabets quickly and without lengthy trial and error.
3. From Sect. 8, we improved the data transfer rate of this proposed system by increasing the number of bits transferred per symbol by 200% for the 256-beam alphabet. The 256-beam alphabet achieved $> 98\%$ accuracy when trained on 700 images per beam, or 178,500 images total. The alphabet achieved a BER of 1,680 incorrect messages per 100,000 messages sent. While this BER is higher than practically usable, further research into optimizing the CNN for classification of LG beams would likely decrease this number even more. Additionally, the simulation results (weak turbulence) indicate the alphabet would achieve an even BER when propagated in more realistic turbulence.
4. From Sect. 8, we improved data rate even further with the 1024-beam alphabet, increasing the number of bits transferred per symbol by 250%. The alphabet achieved $> 91\%$ accuracy when trained on 500 images per beam, or 511,500 total images. The alphabet recorded a BER of 8,740 incorrect messages per 100,000 messages sent. However, the CNN's ability to learn and classify such small variations in alphabet symbols, even in the presence of

strong turbulence, suggests the alphabet would perform even better in more realistic turbulence.

5. From Sect. 9, we discover the CNN struggles when beam testing data is misaligned from beam training data. The CNN is capable of learning a high-level of variation, but only if it has seen the images during the training process. This tells us how essential it is to train your CNN with a wide variety and large quantity of images in order to achieve high classification accuracies. Additionally, we record that beam misalignment can be mitigated through training dataset augmentation.

Section 11B: Lessons Learned

The biggest lessons learned in this project were how to handle large datasets in an efficient manner. Each data collection contained $1100 \times N$ beams, each of 320×320 pixels. This meant 281,600 images for each of the 256-beam experiments, and 1,126,400 images for the 1024-beam experiment! If I wasn't thoughtful, the processing of these images would take hours, and sometimes came out incorrect. This required streamlining of my MATLAB code, as well as explaining why I placed such an emphasis on synchronization of the SLM and camera for data collection.

Section 11C: Future Work and Ongoing Research

There is a plethora of ongoing research related to this project. I had the distinct pleasure of being able to assist several of Prof. Avramov-Zamurovic's research students with their projects. MIDN 2/C John McCarthy will be interning at NRL this summer, and completed two semesters of independent research this year, helping to make the RB phase screen split-step simulation. While John did most of the work, I was able to help him with his code as he based it on the Nikishov split-step simulation I prepared in Fall 2021. I was also able to get to know MIDN 2/C Jack Thomas, who will be propagating a Gaussian beam through my Nikishov split-step simulation in order to investigate whether the distortions of the Gaussian beam due to varying levels of turbulence is characterizable by the CNN. For the hybrid work, I was able to help MIDN 3/C Taher Telfah and MIDN 4/C Aidan Hogan with their Introduction to Research class as they generated and propagated LG beams through the hybrid simulated and experimental environment. Finally, I was able to present my work-in-progress Trident to Prof. Avramov-Zamurovic's EW412 class, and two students from that class, MIDN 2/C Fjordia Akhtar and MIDN 2/C Toby Kim, have signed on with Prof. Avramov-Zamurovic to complete independent work regarding laser propagation in the maritime environment, highlighting the current relevance and excitement around this work and related topic. One of the best parts of completing this Trident Scholar project was being able to assist all of these incredibly talented students as they explored their own projects.

While this Trident Project was a very good start, there is lots of research still to do in order to optimize and improve this communication system. This summer I hope to submit my prepared manuscript regarding the selection basis beams for publishing, as well as preparing 2-3 more manuscripts on the impact of experimental RB turbulence on LG beams, beam misalignment, and a wholistic and comparative review of split-step simulation. Additionally, I am interested in exploring open-loop adaptive optics to better our system. The principle of this topic would be to

use the known phase distortions, which can be extracted easily from simulation, to improve the beam resiliency through turbulence by applying adaptive phase changes at the transmitter. The system would be open loop, meaning it does not use real-time data to adjust for the turbulence. While this often makes changes less effective, open loop makes sense in regard to this problem because it is faster, and turbulence is a difficult and complex thing to characterize quickly.

Section 12: References

- [1] T. Doster and A. T. Watnik, "Machine learning approach to OAM beam demultiplexing," *Applied Optics*, vol. 56, no. 12, pp. 3386-3396, 2017.
- [2] S. Avramov-Zamurovic, C. Nelson and J. M. Esposito, "Effects of underwater optical turbulence on light carrying orbital angular momentum and its classification using machine learning," *Journal of Modern Optics*, 2021.
- [3] X. Yi, Z. Li and Z. Liu, "Underwater optical communication performance for laser beam propagation through weak oceanic turbulence," *Applied Optics*, vol. 54, no. 6, pp. 1273-1278, 2015.
- [4] P. Lacovara, "High-bandwidth underwater communications," *Technology Society Journal*, vol. 42, pp. 93-102, 2008.
- [5] S. Arnon, "Underwater optical wireless communication network," *Optical Engineering*, vol. 49, 2010.
- [6] H. Kaushal and G. Kaddoum, "Underwater optical wireless communication," *IEEE Access*, vol. 4, pp. 1518-1547, 2016.
- [7] Z. Zeng, S. Fu, H. Zhang, Y. Dong and J. Cheng, "A survey of underwater optical wireless communications," *IEEE communication surveys and tutorials*, vol. 19, no. 1, pp. 2014-238, 2016.
- [8] H. Oubei, C. Shen, A. Kammoun, E. Zedini, K. Park, X. Sun and G. Liu, "Light based underwater wireless communications," *Japanese Journal of Applied Physics*, vol. 57, 2018.
- [9] I. F. Akyildiz, D. Pompili and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 33, pp. 255-256, 2005.
- [10] D. e. al, "Land and undersea field testing of very low frequency RF antennas and loop transceivers," *US Gov. Technical Report*, 2017.
- [11] G. Gbur, *Singular Optics*, CRC Press/Taylor and Francis, 2016.
- [12] R. L. Nowack, "A tale of two beams: an elementary overview of Gaussian beams and Bessel beams," *Study of Geophysics*, vol. 56, pp. 355-372, 2012.

- [13] S. Avramov-Zamurovic, A. T. Watnik, J. R. Lindle and K. P. Judd, "Designing laser beams carrying OAM for a high-performance underwater communication system," *Journal of the Optical Society of America*, 2020.
- [14] W. A. Jarrett, S. Avramov-Zamurovic and J. M. Esposito, "Machine learning-based design of structured laser light for improved data transfer rate in underwater wireless communication," Trident Scholar Proposal, 2021.
- [15] "Matlab Academy," Mathworks, 2020. [Online]. Available: <https://matlabacademy.mathworks.com/R2020a/portal.html?course=mldl>. [Accessed 2021].
- [16] G. e. al, Deep Learning, MIT Press, 2016.
- [17] S. Avramov-Zamurovic and e. al, "Machine-learning-aided classification of beams carrying orbital angular momentum propagated in highly turbid water," *Journal of Optical Society of America*, 2020.
- [18] X. Cui, X. Yin, H. Chang, H. Liao, X. Chen, X. Xin and Y. Wang, "Experimental study of machine-learning-based orbital angular momentum shift keying decoders in optical underwater channels," *Optical Communications*, vol. 452, pp. 116-123, 2019.
- [19] Neary, Machine learning enhanced free-space and underwater OAM optical communiations, Utah State University, 2020.
- [20] P. L. Neary, Machine learning enhanced free-space and underwater OAM optical communiations, Utah State University, 2020.
- [21] A. E. e. a. Willner, "Underwater optical communications using orbital angular momentum-based spatial division multiplexing," *Optics Communication*, 2017.
- [22] C. Nelson, S. Avramov-Zamurovic, R. Malek-Madani, O. Korotkova, R. Sova and e. al., "Measurements and comparison of the probability density and covariance functions of laser beam intensity fluctuations in a hot-air turbulence emulator with the maritime atmospheric environment," in *Proceedings of SPIE*, San Diego, 2012.
- [23] S. Avramov-Zamurovic, C. Nelson, S. Guth, O. Korotkova and R. Malek-Madani, "Experimental study of electromagnetic Bessel-Gaussian Schell Model beams propagating in a turbulent channel," *Optics Communications*, vol. 359, pp. 207-215, 2016.
- [24] L. C. Andrews and R. L. Phillips, Laser Beam Propagation through Random Media, 2nd Edition, SPIE, 2015.
- [25] S. Avramov-Zamurovic, C. Nelson and J. M. Esposito, "Experimentally evaluating beam scintillation and vortex structure as a function of topological charge in underwater optical turbulence," *Submitted to Optics Communication*, 2021.

- [26] Meadowlark Optics, "User Manual 1920x1152 XY Phase Series Spatial Light Modulator With PCIe Controller," Frederick, CO.
- [27] Fastec Imaging, "TS3, TS4, and TS5 High Speed Cameras Operator's Manual," Fastec Imaging Corporation, San Diego, CA, 2015.
- [28] J. D. Schmidt, Numerical Simulation of Optical Wave Propagation, Bellingham, WA: SPIE Press, 2010.
- [29] O. Korotkova, "Light propagation in a turbulent ocean," *Progress in Optics*, vol. 64, pp. 1-43, 2019.
- [30] L. Lu, X. Ji and Y. Baykal, "Wave structure function and spatial coherence radius of plane and spherical waves propagating through oceanic turbulence," *Optics Express*, vol. 22, no. 22, 2014.
- [31] I. Toselli and S. Gladysz, "On the general equivalence of the Fried parameter and coherence radius for non-Kolmogorov and oceanic turbulence," *OSA Continuum*, vol. 2, no. 1, 2019.
- [32] B. L. McGlamery, "Restoration of turbulence-degraded images," *J. Opt. Soc. Am.*, vol. 57, no. 3, pp. 293-297, 1967.
- [33] V. V. Nikishov and V. I. Nikishov, "Spectrum of turbulent fluctuations of the sea-water refraction index," *International Journal of Fluid Mechanics Research*, vol. 27, pp. 82-98, 2000.
- [34] R. Lane, A. Glindemann and J. C. Dainty, "Simulation of a Kolmogorov phase screen," *Waves in Random Media*, vol. 2, pp. 209-224, 1992.
- [35] N. Farwell and O. Korotkova, "Multiple phase-screen simulation of oceanic beam propagation," in *Proceedings of SPIE, Laser Communication and Propagation through the Atmosphere and Ocean*, San Diego, CA, 2014.
- [36] Y. Ren and e. al, "Orbital angular momentum-based space division multiplexing for high-capacity underwater optical communications," *Sci. Rep.*, vol. 6, 2016.
- [37] B. Cochenour and e. al., "Propagation of modulated optical beams carrying orbital angular momentum (OAM) in turbid water," *Applied Optics*, 2016.
- [38] A. Krizhevsky, I. Sutskever and G. E. and Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems 25*, vol. 25, pp. 1097-1105, 2012.
- [39] S. Avramov-Zamurovic and e. al., "Machine-learning-aided classification of beams carrying orbital angular momentum propagated in highly turbid water," *Journal of Optical Society of America*, 2020.

- [40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, 2015.
- [41] S. Lohani, E. M. Knutson, M. O'Donnell, S. Huver and R. T. Glasser, "On the use of deep neural networks in optical communications," *Applied Optics*, vol. 57, no. 15, pp. 4180-4190, 2018.
- [42] X.-Z. Cui, X.-L. Yin, H. Chang and e. al., "Analysis of an adaptive orbital angular momentum shift keying decoder based on machine learning under oceanic turbulence channels," *Optics Communication*, vol. 429, pp. 138-143, 2018.
- [43] Z. Wang, M. I. Dedo, K. Guo and e. al., "Efficient recognition of the propagated orbital angular momentum modes in turbulences with the convolutional neural network," *IEEE Photonics*, vol. 11, no. 3, 2019.
- [44] P. L. Neary, A. T. Watnik, K. P. Judd, J. R. Lindle and N. S. Flann, "CNN classification architecture study for turbulent free-space and attenuated underwater optical OAM communications," *Applied Science*, vol. 10, no. 24, 2020.
- [45] W. A. Jarrett, S. Avramov-Zamurovic, C. Nelson, J. M. Esposito and M. Hyde, "Neural network classification of structured light in optical turbulence," in *Proceedings of SPIE*, 2021.
- [46] S. Basu and M. Hyde, "Computational approaches for generating electromagnetic Gaussian Schell-model sources," *Optics Express*, vol. 22, no. 26, 2014.
- [47] G. Nootz, E. Jarosz, F. D. Dalgleish and W. Hou, "Quantification of optical turbulence in the ocean and its effects on beam propagation," *Applied Optics*, vol. 55, no. 31, pp. 8813-8820, 2016.
- [48] K. P. Judd, S. Avramov-Zamurovic, S. Matt, R. A. Handler, J. R. Lindle, J. M. Esposito and W. A. Jarrett, "Propagation of laser beams carrying orbital angular momentum through simulated optical turbulence in Rayleigh-Bénard convection," in *Proceedings of SPIE*, 2021.

Appendix A: MATLAB Code Written for Trident Scholar Project

Section 1: LGbeamsforML.m – script to generate and save SLM beam screens

```

%% Generation of theoretical LG Beams SLM screens
clearvars; clc; close all
tic

% Code library:
% LGBeamCombinations.m
% LaguerreGen.m
% Interp2_Stack.m
% Form_Grating.m
% SLM_bmp_J0Method.m
% SLM_bmp_J1Method.m
% Numerical_Grating_Height_Inversed.mat

% LC-R 720 Reflective LCOS SLM
% Resolution: 1280 x 768
% Pixel Pitch: 20 µm
% Fill Factor: 92 %
% Active Area 25.6 x 15.4 mm (1.18? Diagonal)
% Addressing 8 Bit (256 Grey Levels)
% Signal Formats DVI - WXGA Resolution
% Frame Rate 60Hz / 180 Hz
% Hen Green 1.5 mW 543 nm

%% Import images for annotation
% Im(:,:,1) = imread('Beam_Imgs_for_Ant/Beam_1/img1.png');
% Im(:,:,2) = imread('Beam_Imgs_for_Ant/Beam_8/img1.png');
% Im(:,:,3) = imread('Beam_Imgs_for_Ant/Beam_10/img1.png');
% Im(:,:,4) = imread('Beam_Imgs_for_Ant/Beam_14/img1.png');
% Im(:,:,5) = imread('Beam_Imgs_for_Ant/Beam_9/img1.png');
% Im(:,:,6) = imread('Beam_Imgs_for_Ant/Beam_11/img1.png');
% Im(:,:,7) = imread('Beam_Imgs_for_Ant/Beam_15/img1.png');
% Im(:,:,8) = imread('Beam_Imgs_for_Ant/Beam_12/img1.png');
% Im(:,:,9) = imread('Beam_Imgs_for_Ant/Beam_2/img1.png');
% Im(:,:,10) = imread('Beam_Imgs_for_Ant/Beam_4/img1.png');
% Im(:,:,11) = imread('Beam_Imgs_for_Ant/Beam_13/img1.png');
% Im(:,:,12) = imread('Beam_Imgs_for_Ant/Beam_3/img1.png');
% Im(:,:,13) = imread('Beam_Imgs_for_Ant/Beam_5/img1.png');
% Im(:,:,14) = imread('Beam_Imgs_for_Ant/Beam_6/img1.png');
% Im(:,:,15) = imread('Beam_Imgs_for_Ant/Beam_7/img1.png');
%
% % for i = 1:15
% % Im(:,:,i) = imresize(Im(:,:,i), [256,256]);
% % end

%% All Parameters
alpha_ord = [0 1 0 1]; % basis beam set ALPHA
alpha_top = [1 4 -6 9];
echo_ord = [0 1 2 3 0 1]; % basis beam set ECHO
echo_top = [1 9 7 9 8 1];
foxtrot_ord = [0 1 0 1 1 2 2 3]; % basis beam set FOXTROT
foxtrot_top = [1 4 -6 9 -8 -3 7 5];

strong_ord = [0 1 2 3 0 1 2 3];
strong_top = [1 9 -7 9 -8 1 5 6];

weak_ord = [0 0 0 0 1 1 1 1];
weak_top = [1 2 -3 4 1 -2 3 4];

med_ord = [0 1 2 3 0 1 2 3];
med_top = [1 5 -3 5 -4 1 2 3];

zeta_ord = [0 1 2 3 0 1 2 3 0 1];
zeta_top = [1 9 -7 9 -8 1 5 6 4 -5];

```

```

% ===== WHAT YOU CHANGE ===== %
wvl = 635e-9; % [m] - red HeNe laser
k = 2*pi/wvl; % [m^-1] - wave number
Grating_Period = 8; % spacing between modes
w0 = 1e-3; % [m] - beam waist
ord = zeta_ord; % order of Laguerre polynomial (first)
top = zeta_top; % topological charge (second)
experimentFolder = sprintf('1024_Set'); % overarching experiment
beamSetFolder = sprintf('Test'); % screens for specific set of basis beams

SLM_PixelsX = 1920; SLM_PixelsY = 1152; % Big USNA
% SLM_PixelsX = 1280; SLM_PixelsY = 768; % Holoeve at NRL
% SLM_PixelsX = 800; SLM_PixelsY = 800; % USNA

% SLM_Pitch_y = 10.7e-3 / 1152; % Big USNA
% SLM_Pitch_x = 17.6e-3 / 1920;
SLM_Pitch = 9.2e-6;
% SLM_Pitch = 20e-6; % Holoeve at NRL
% SLM_Pitch = 24e-6; % USNA
% SLM_Pitch_x = SLM_Pitch;
% SLM_Pitch_y = SLM_Pitch;

% ===== %

% takes care of folders
cd(experimentFolder);
mkdir(beamSetFolder);
cd(beamSetFolder);
scrnFolder = sprintf('SLM_Screens'); % folder to put screens in
mkdir(scrnFolder);
cd ../../

%% Create Grids
SLM_Lx = SLM_PixelsX*SLM_Pitch; SLM_Ly = SLM_PixelsY*SLM_Pitch;

Nx = SLM_PixelsX; Ny = SLM_PixelsY;
Nx1 = Nx; % resolution in x direction
Ny1 = Ny; % resolution in y direction
delta_y = SLM_Pitch; % pixel size in mm - comes from 25.6 mm / 1280
delta_x = SLM_Pitch; % pixel size in mm - comes from 15.4 mm / 768
% lambda = 633e-9; % mm Wavelength of HeNe
x = Grating_Period; % Grating period

k_y = 2*pi / (x*delta_y); % Blaze angle in y direction in radian
k_x = 0; % vertical fork
X_1d = (-0*delta_x) + linspace(-1/2,1/2,Nx)*(Nx*delta_x);
Y_1d = (0*delta_y) + (linspace(-1/2,1/2,Ny))*(Ny*delta_y);
[X_2d,Y_2d] = meshgrid(X_1d,Y_1d); % Cartesian coordinates of the pixels
[THETA,RHO] = cart2pol(X_2d,Y_2d); % Polar coordinate of the pixels

% polar coordinates for SLM
[xslm,yslm] = meshgrid((-SLM_PixelsX/2:SLM_PixelsX/2-1)*SLM_Pitch, (-SLM_PixelsY/2:SLM_PixelsY/2-1)*SLM_Pitch);
[phislm,rhoslm] = cart2pol(xslm,yslm);

%% Laguerre-Gaussian Parameters - this creates the beams
% The SLM phase plate is at z = 0
z = 0;

zR = pi*w0^2 / wvl; % [m] - Rayleigh range
wz = w0 * sqrt(1+(z/zR)^2); % [m] - effective radius of beam at distance z

n = length(top);
beamTotal = 2^n - 1;
montageN = sqrt(beamTotal+1);

for i = 1:n
    bin(:,i) = zeros([1 n]);
end

```

```

for i = 1:n
    clg(i) = sqrt(2*factorial(ord(i))/(pi*factorial(ord(i)+abs(top(i))))); % Laguerre bit
    LG(:,i,i) =
    clg(i)/wz*(rhoslm*sqrt(2)/wz).^abs(top(i)).*polyval(LaguerreGen(ord(i),abs(top(i))),2*rhoslm.^2/wz^2)...
        .*exp(-rhoslm.^2/wz^2).*exp(1j*k*rhoslm.^2*z/(2*(z^2+zR^2))).*exp(-1j*( 2*ord(i)+abs(top(i))+1
        )*atan(z/zR))...
        .*exp(1j*top(i)*phislm).*exp(-1j*k*z).*exp(-1j*atan(z/zR));
    bin(:,i,i) = bin(:,i,i)+1;
end

%% Generate all possible combinations of n base beams
[U, beam, beamBin] = LGBeamCombinations(n, beamTotal, LG, bin); % function returns 2^n Us (all combos)

%% Generate and save basic Gaussian beam
imgNum = 0;
pos = [.6 .4 .5 .5];

ord_0 = 0;
top_0 = 0;
clg_0 = sqrt(2*factorial(ord_0)/(pi*factorial(ord_0+abs(top_0))))); % Laguerre bit
U_0 = clg_0/wz*(rhoslm*sqrt(2)/wz).^abs(top_0).*polyval(LaguerreGen(ord_0,abs(top_0)),2*rhoslm.^2/wz^2)...
        .*exp(-rhoslm.^2/wz^2).*exp(1j*k*rhoslm.^2*z/(2*(z^2+zR^2))).*exp(-1j*(
        2*ord_0+abs(top_0)+1)*atan(z/zR))...
        .*exp(1j*top_0*phislm).*exp(-1j*k*z).*exp(-1j*atan(z/zR));

U_beam = U_0;
% Theoretical beam image (theoretical close image)
figure(1); imagesc(abs(U_beam)); % intensity distribution
title(sprintf('Intensity Dist of Beam %d', imgNum))

% Save for montage
set(gca,'position',[0 0 1 1],'units','normalized');
% F = getframe(gcf);
% [I1(:, :, imgNum), Map] = frame2im(F);
% I1(:, :, imgNum) = imresize(I1(:, :, imgNum), [277,277]);

% Phase intensity distribution (Phase Change Image)
figure(2); imagesc(angle(U_beam)); % phase angle
title(sprintf('Phase Angle of Beam %d', imgNum))
% set(gca,'position',[0 0 1 1],'units','normalized');
% F = getframe(gcf);
% [I(:, :, imgNum), Map] = frame2im(F);

% BMP Image (SLM Screen)
[SLM_image, SLM_phase] = SLM_bmp_J1Method(U_beam.*exp(1i*(X_2d.*k_x+Y_2d.*k_y))); %Generates the SLM
hologram
SLM_phase = double(SLM_phase);
xz = double((SLM_phase-min(min(SLM_phase))));
SLM_phase = xz./max(max(xz))*2*pi;
% figure(3); imagesc(abs(SLM_phase(SLM_PixelsY/2-64 : SLM_PixelsY/2+64, SLM_PixelsX/2-64 :
SLM_PixelsX/2+64)));
% figure(4); imagesc(abs(SLM_phase(1:SLM_PixelsY, 1:SLM_PixelsX))); % unlocalized

figure(4); imagesc(abs(SLM_phase)); % BMP map image

% **Creates and saves BMP file**
SLM_phase = SLM_phase/max(max(SLM_phase))*255;
cd(experimentFolder); cd(beamSetFolder); cd(scrnFolder);
scrnName = sprintf('Beam_%04d.bmp', imgNum);
imwrite(uint8(SLM_phase),scrnName,'bmp');
cd ../../..

%% Generate figures
imgNum = 1;
pos = [.6 .4 .5 .5];
% b = de2bi(d)
for kn = 1:beamTotal
    U_beam = U(:, :, kn);

```

```

% Theoretical beam image (theoretical close image)
figure(1); imagesc(abs(U_beam)); % intensity distribution
title(sprintf('Intensity Dist of Beam %d', kn))
% axis([1 1280 -256 1024])
% % Binary annotation
% delete(findall(gcf,'type','text'));
% txt = text(153, 45, num2str(beamBin(:,kn)));
% txt.FontSize = 36; txt.Color = 'w'; txt.FontWeight = 'bold'; txt.FontName = 'TimesNewRoman';
% Save for montage
set(gca,'position',[0 0 1 1],'units','normalized');
F = getframe(gcf);
[I1_0(:, :, :, imgNum), Map] = frame2im(F);
I1(:, :, :, imgNum) = imresize(I1_0(:, :, :, imgNum), [277,277]);

% % Actual beam image (photographs)
% figure(12); imshow(Im(:, :, kn), 'border', 'tight');
% % Binary annotation
% delete(findall(gcf,'type','text'));
% txt = text(140, 45, num2str(beamBin(:,kn)));
% txt.FontSize = 16; txt.Color = 'y'; txt.FontWeight = 'bold'; txt.FontName = 'TimesNewRoman';
% % Save for montage
% set(gca,'position',[0 0 1 1], 'units', 'normalized');
% F = getframe(gcf);
% [I2(:, :, :, imgNum), Map] = frame2im(F);

% Phase intensity distribution (Phase Change Image)
figure(2); imagesc(angle(U_beam)); % phase angle
title(sprintf('Phase Angle of Beam %d', kn))
% set(gca,'position',[0 0 1 1],'units','normalized');
% F = getframe(gcf);
% [I(:, :, :, imgNum), Map] = frame2im(F);

% All of this is looking at characterizing SLM
% SLM grating (Far-Field Image)
[SLM_Grating, SLM_Phase_Fix] = Form_Grating({SLM_PixelsX SLM_PixelsY SLM_Pitch}, Grating_Period, ...
abs(U_beam)/max(max(abs(U_beam))), 'bothxy', 'theory');

temp = exp(1j*2*pi*SLM_Grating).*exp(1j*angle(U_beam));
Uslm = temp;

% Far field Image

% steer = 2*pi/Grating_Period*xslm/SLM_Pitch + 2*pi/Grating_Period*yslm/SLM_Pitch;
% Nsim = 4096*2;
% System_Aberration = zeros(SLM_PixelsY, SLM_PixelsX);
% padSLM = padarray(Uslm.*exp(1j*2*pi*System_Aberration), [Nsim/2-SLM_PixelsY/2 Nsim/2-
SLM_PixelsX/2]); % ----- %
% figure(6); imagesc(angle(USAZpadslm)); title('angle Uslm padded with system aberrations')
% far_field_img = fftshift(fft2(padSLM));
% figure(7); imagesc(abs(far_field_img)); title('Far-Field Image')

% BMP Image (SLM Screen)
[SLM_image, SLM_phase] = SLM_bmp_J1Method(U_beam.*exp(1i*(X_2d.*k_x+Y_2d.*k_y))); %Generates the SLM
hologram
SLM_phase = double(SLM_phase);
xz = double((SLM_phase-min(min(SLM_phase))));
SLM_phase = xz./max(max(xz))*2*pi;
% figure(3); imagesc(abs(SLM_phase(SLM_PixelsY/2-64 : SLM_PixelsY/2+64, SLM_PixelsX/2-64 :
SLM_PixelsX/2+64)));
% figure(4); imagesc(abs(SLM_phase(1:SLM_PixelsY, 1:SLM_PixelsX))); % unlocalized

figure(4); imagesc(abs(SLM_phase)); % BMP map image

% axis([1 1280 -256 1024])

% title(sprintf('SLM Screen of Beam %d', kn))
% str = sprintf('%d,%d', p(kn), ell(kn));
% annotation('textbox', dim, 'String', str, 'FitBoxToText', 'on', 'Color', 'k');

```

```

%     set(gca,'position',[0 0 1 1],'units','normalized');
%     F3 = getframe(gcf);
%     [I3(:,:,imgNum), Map] = frame2im(F3);

% **Creates and saves BMP file**
    SLM_phase = SLM_phase/max(max(SLM_phase))*255;
    cd(experimentFolder); cd(beamSetFolder); cd(scrnFolder);
    scrnName = sprintf('Beam_%04d.bmp', kn);
    imwrite(uint8(SLM_phase),scrnName,'bmp');
    cd ../../..

imgNum = imgNum+1;
% pause();
end

%% Display Montage
figure; montage(I1, 'Size', [montageN montageN]);
set(gca,'position',[0 0 1 1],'units','normalized');

figure; montage(I1(:,:,1:10), 'Size', [2 5]);
title('Basis Beams');

% figure; montage(I2, 'Size', [3 5]);
% set(gca,'position',[0 0 1 1],'units','normalized');

% figure; montage(I3, 'Size', [NaN 4]);
% set(gca,'position',[0 0 1 1],'units','normalized');

elapsed_minutes = floor(toc/60);
elapsed_seconds = round(((toc/60) - floor(toc/60))*60, 0);
fprintf('Total time to generate screens: %d:%d\n', elapsed_minutes, elapsed_seconds)

```

Section 2: LGBeamCombinations.m – function to create LG beam combinations and encode information

```
function [U, beam, beamBin] = LGBeamCombinations(n, beamTotal, LG, bin)
% combines LG beams and provides corresponding binary coding
beam = 1;
    for i = 1:n
        U(:, :, beam) = LG(:, :, i);
        beamBin(:, :, beam) = bin(:, :, i);
        fprintf('Beam %d\n', beam);
        beam = beam + 1;
    end
if (beam == beamTotal+1)
    fprintf('Done\n');
    return;
end
    for i = 1:n-1
        for j = i+1:n
            U(:, :, beam) = LG(:, :, i) + LG(:, :, j);
            beamBin(:, :, beam) = bin(:, :, i) + bin(:, :, j);
            fprintf('Beam %d\n', beam);
            beam = beam + 1;
        end
    end
if (beam == beamTotal+1)
    fprintf('Done\n');
    return;
end
    for i = 1:n-2
        for j = i+1:n-1
            for k = j+1:n
                U(:, :, beam) = LG(:, :, i) + LG(:, :, j) + LG(:, :, k);
                beamBin(:, :, beam) = bin(:, :, i) + bin(:, :, j) + bin(:, :, k);
                fprintf('Beam %d\n', beam);
                beam = beam+1;
            end
        end
    end
if (beam == beamTotal+1)
    fprintf('Done\n');
    return;
end
    for i = 1:n-3
        for j = i+1:n-2
            for k = j+1:n-1
                for l = k+1:n
                    U(:, :, beam) = LG(:, :, i) + LG(:, :, j) + LG(:, :, k) + LG(:, :, l);
                    beamBin(:, :, beam) = bin(:, :, i) + bin(:, :, j) + bin(:, :, k) + bin(:, :, l);
                    fprintf('Beam %d\n', beam);
                    beam = beam + 1;
                end
            end
        end
    end
if (beam == beamTotal+1)
    fprintf('Done\n');
    return;
end
    for i = 1:n-4
        for j = i+1:n-3
            for k = j+1:n-2
                for l = k+1:n-1
                    for m = l+1:n
                        U(:, :, beam) = LG(:, :, i)+LG(:, :, j)+LG(:, :, k)+LG(:, :, l)+LG(:, :, m);
                        beamBin(:, :, beam) = bin(:, :, i) + bin(:, :, j) + bin(:, :, k) + bin(:, :, l) +
bin(:, :, m);
                        fprintf('Beam %d\n', beam);
                        beam = beam + 1;
                    end
                end
            end
        end
    end
```

```

        end
    end
end
end
end
if (beam == 2^n)
    fprintf('Done\n');
    return;
end
for i = 1:n-5
    for j = i+1:n-4
        for k = j+1:n-3
            for l = k+1:n-2
                for m = l+1:n-1
                    for o = m+1:n
                        U(:, :, beam) = LG(:, :, i) + LG(:, :, j) + LG(:, :, k) + LG(:, :, l) + LG(:, :, m) ...
                            + LG(:, :, o);
                        beamBin(:, :, beam) = bin(:, :, i) + bin(:, :, j) + bin(:, :, k) + bin(:, :, l) +
bin(:, :, m) ...
                            + bin(:, :, o);
                        fprintf('Beam %d\n', beam);
                        beam = beam + 1;
                    end
                end
            end
        end
    end
end
if (beam == beamTotal+1)
    fprintf('Done\n');
    return;
end
for i = 1:n-6
    for j = i+1:n-5
        for k = j+1:n-4
            for l = k+1:n-3
                for m = l+1:n-2
                    for o = m+1:n-1
                        for q = o+1:n
                            U(:, :, beam) = LG(:, :, i) + LG(:, :, j) + LG(:, :, k) + LG(:, :, l) + LG(:, :, m) ...
                                + LG(:, :, o) + LG(:, :, q);
                            beamBin(:, :, beam) = bin(:, :, i) + bin(:, :, j) + bin(:, :, k) + bin(:, :, l) +
bin(:, :, m) ...
                                + bin(:, :, o) + bin(:, :, q);
                            fprintf('Beam %d\n', beam);
                            beam = beam + 1;
                        end
                    end
                end
            end
        end
    end
end
if (beam == beamTotal+1)
    fprintf('Done\n');
    return;
end
for i = 1:n-7
    for j = i+1:n-6
        for k = j+1:n-5
            for l = k+1:n-4
                for m = l+1:n-3
                    for o = m+1:n-2
                        for q = o+1:n-1
                            for r = q+1:n
                                U(:, :, beam) = LG(:, :, i) + LG(:, :, j) + LG(:, :, k) + LG(:, :, l) + LG(:, :, m) ...
                                    + LG(:, :, o) + LG(:, :, q) + LG(:, :, r);
                                beamBin(:, :, beam) = bin(:, :, i) + bin(:, :, j) + bin(:, :, k) + bin(:, :, l)
+ bin(:, :, m) ...

```



```

        end
    end
end
if (beam == beamTotal+1)
    fprintf('Done\n');
    return;
end

```

Section 3: Beamset_Corr.m – script to correlate and check alphabet

```
%% Check Correlation Between Generated Beam Set
```

```
% Function Library:
```

```
% genLG.m - makes LG beam
```

```
clearvars; clc
```

```
%% Parameters
```

```
alpha_ord = [0 1 0 1]; % parameters for baseline "ALPHA" set of beams
```

```
alpha_top = [1 4 -6 9];
```

```
echo_ord = [0 1 0 1 2 2]; % basis beam set ECHO
```

```
echo_top = [1 4 -6 9 -8 -3];
```

```
foxtrot_ord = [0 1 0 1 1 2 2 3]; % basis beam set FOXTROT
```

```
foxtrot_top = [1 4 -6 9 -8 -3 7 5];
```

```
strong_ord = [0 1 2 3 0 1 2 3];
```

```
strong_top = [1 9 -7 9 -8 1 5 6];
```

```
weak_ord = [0 0 0 0 1 1 1 1];
```

```
weak_top = [1 2 -3 4 1 -2 3 4];
```

```
med_ord = [0 1 2 3 0 1 2 3];
```

```
med_top = [1 5 -3 5 -4 1 2 3];
```

```
zeta_ord = [0 1 2 3 0 1 2 3 0 1];
```

```
zeta_top = [1 9 -7 9 -8 1 5 6 4 -5];
```

```
% ===== THIS IS WHAT YOU CHANGE ===== %
```

```
N = 2^8; % number of grid points per side
```

```
D1 = 0.1; % [m] - side length of source plane
```

```
D2 = D1; % [m] - side length of observation plane
```

```
delta_1 = D1/N; % [m] - grid spacing at source
```

```
delta_n = D2/N; % [m] - grid spacing at receiver
```

```
wvl = 633e-9; % [m] - wavelength
```

```
w0 = 1e-2; % [m] - beam waist radius
```

```
ord = zeta_ord; % order of Laguerre polynomial of LG beams - can be given as an array of basis beams  
(combinations will be made automatically)
```

```
top = zeta_top; % topological charge of transmitted LG beams
```

```
% ===== %
```

```
k = (2*pi) / wvl; % [rad/m] - wavenumber
```

```
[x1, y1] = meshgrid((-N/2 : N/2-1)*delta_1); % grid size: L1 x L1 [m]
```

```
[theta1, r1] = cart2pol(x1, y1);
```

```
% Generate all possible combinations of n base beams
```

```
n_beam = length(top);
```

```
beamTotal = 2^n_beam - 1;
```

```
LG = zeros(N,N,n_beam);
```

```
for i = 1:n_beam
    bin(:,i) = zeros([1 n_beam]);
end
```

```
for j = 1:n_beam
    LG(:,j) = genLG(top(j), ord(j), wvl, x1, y1, w0); % topological charge, order of LG polynomial,  
grid, beam waist
end
```

```
[U, beam, beamBin] = LGBeamCombinations(n_beam, beamTotal, LG, bin); % generates array of complex  
amplitude of all input beams
```

```

%% Check Correlation
idx = 0; corrSum = 0;
corrMatrix = zeros(beamTotal, beamTotal);
for i = 1:beamTotal
    for j = i:beamTotal
        corrMatrix(i,j) = corr2(abs(U(:,i)), abs(U(:,j)));
        corrSum = corrMatrix(i,j) + corrSum;
        idx = idx + 1;
    end
end
avgCorr = corrSum / idx;
avgCorrStr = sprintf('Average Correlation: %.2f', avgCorr);
figure; imagesc(corrMatrix); colorbar
% title(avgCorrStr);

```

Section 4: genLG.m – function to make a LG beam from inputed topological charge m and order of Laguerre polynomial n (saves time when writing new code)

```

function U = genLG(top, ord, lambda, x, y, D)
% WAJ
% top - topological charge of beam
% ord - order of Laguerre polynomial
% lambda - wvl of optical light
% N - size of grid
% uses function LaguerreGen.m
% test - top = 1; ord = 0; lambda = 1e-6; N = 512;

[phi,rho] = cart2pol(x, y);

% Generate Laguerre-Gaussian Beams
z = 0; % created at source plane
w0 = D; % beam waist
k = (2*pi)/lambda; % 1/m - wave number

zr = pi*w0^2 / lambda; % Rayleigh range
wz = w0 * sqrt(1+(z/zr)^2); % effective radius of beam at distance z

% These are the full propagation formulas that describe the beam at dist z
clg = sqrt(2*factorial(ord)./(pi*factorial(ord+abs(top)))); % Laguerre bit

U = clg/wz*(rho*sqrt(2)/wz).^abs(top).*polyval(LaguerreGen(ord,abs(top)),2*rho.^2/wz^2)...
    .*exp(-rho.^2/wz^2).*exp(1j*k*rho.^2*z/(2*(z^2+zr^2))).*exp(-1j*( 2*ord+abs(top)+1 )*atan(z/zr))...
    .*exp(1j*top*phi).*exp(-1j*k*z).*exp(-1j*atan(z/zr)); % complex amplitude of beam

%figure; imagesc(abs(U))

```

Section 5: Tiff_Processor.m – script to process raw camera data, works best for large camera files where saved images are placed in “img####” folders

```
%% Tiff_Processor for Experimental work

% Purpose: splits 1 TIFF file into multiple folders
% pre-processes images for CNN
% tic

clearvars; close all; clc

%% Parameters
% ===== WHAT YOU CHANGE ===== %
experimentFolder = sprintf('1024_Set/1st_half'); % folder for experiment (organizes everything in matlab)
dataFolder = sprintf('1024_11fpb_275fps_20hz_000000'); % this is the big folder everything goes in
CNNFolder = sprintf('For_CNN_64');
numLoops = 100; % number of loops on SLM
cycleFrames = 11; % number of frames per trigger of camera (post cycling)
beamTotal = 513; % number of beams in set
imgStart = 2800; % first image in the set
imgFirst = 2829; % where to start
imgLast = imgFirst+(beamTotal*cycleFrames*numLoops); % where to end
cnn_img_size = 64; % pixels - size of square image for CNN
LOW_IN = 0.08; HIGH_IN = 0.17; LOW_OUT = 0; HIGH_OUT = 1; % parameters to control imadjust
% ===== %

%%
cd(experimentFolder);
cd(dataFolder);

% creates folder for CNN data
mkdir(CNNFolder);
cd(CNNFolder);

for i = 1:beamTotal
    newFolder = sprintf('Beam_%04d', i-1);
    mkdir(newFolder);
end

cd ..

camNum = 0;
idxImg = imgFirst;

% Test cropping first

camImgFolder = sprintf('img%04d', camNum);
dataFileName = sprintf('%07d.tif', idxImg);
beamFileName = fullfile(camImgFolder,dataFileName);
beam_px_val = imread(beamFileName); % i needs to be an int here
beam_px_val = uint8(beam_px_val); % convert to uint8 (just like camera)
beamImg = im2gray(beam_px_val); % convert to gray scale
figure; imshow(beam_px_val); % title('Unprocessed')

beamImg = imadjust(beamImg, [LOW_IN; HIGH_IN], [LOW_OUT; HIGH_OUT]); % adjust contrast between beam and
background
[beamImg, crop_dim] = imcrop(beamImg); % crops image, determines setting for all crops
beamImg = imresize(beamImg, [cnn_img_size cnn_img_size]); % resizes image to input for CNN
figure; imshow(beamImg); title('Processed') % show processed image

% pause();

while idxImg < imgLast
    for i = 1:beamTotal
        folderName = sprintf('/Beam_%04d', i-1); % folder to put images in
        folderName = strcat(CNNFolder, folderName);

        j_start = idxImg;
```

```

j_cut = (idxImg+cycleFrames-1);
for j = j_start:j_cut-1

    camImgFolder = sprintf('img%04d', camNum);
    dataFileName = sprintf('%07d.tif', idxImg);
    beamFileName = fullfile(camImgFolder,dataFileName);
    beam_px_val = imread(beamFileName); % i needs to be an int here
    beam_px_val = uint8(beam_px_val); % convert to uint8 (just like camera)
    beamImg = im2gray(beam_px_val); % convert to gray scale
    beamImg = imadjust(beamImg, [LOW_IN; HIGH_IN], [LOW_OUT; HIGH_OUT]); % adjust contrast between
beam and background
    beamImg = imcrop(beamImg, crop_dim); % crops image
    beamImg = imresize(beamImg, [cnn_img_size cnn_img_size]); % resizes image to input for CNN

    imgFileName = sprintf('img%07d.png', j);
    fullFileName = fullfile(folderName, imgFileName);
    imwrite(beamImg, fullFileName);
    fprintf('%s_img%d\n', folderName, j)

    x = (idxImg-(imgStart-1))/30000; % checks to make sure it accesses the right folder
    if (rem(x,1) == 0)
        camNum = camNum+1;
    end

    idxImg = idxImg + 1;
end

% cuts out bad 11th image
x = (idxImg-(imgStart-1))/30000; % checks to make sure it accesses the right folder
if (rem(x,1) == 0)
    camNum = camNum+1;
end
idxImg = idxImg + 1;

end
end

cd ../../..
% elapsed_minutes = floor(toc/60);
% elapsed_seconds = round(((toc/60) - floor(toc/60))*60, 0);
% fprintf('Total time to process images: %d:%d\n', elapsed_minutes, elapsed_seconds)

```

Section 6: CNN.m – script to create and run CNN

```

%% CNN Training and Testing
clearvars; close all; clc
% tic

%% Folder Information
% ===== WHAT YOU CHANGE ===== %
experimentFolder = sprintf('1024_set'); % folder for experiment (organizes everything in matlab)
dataFolder = sprintf('Zeta'); % this is the big folder all the data goes in
imNum = 1000; % number of images per beam
totalBeams = 1023; % total number of beams being classified
trainImg = 500; % number of images per beam used for training
testImg = 500; % number of images used per beam for testing
% ===== %

cnnFolder = sprintf('For_CNN_64');

% access images
cd(experimentFolder);
cd(dataFolder);

%% CNN Set Up
imds = imageDatastore(cnnFolder, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');

```

```

p = trainImg / imNum; % percentage of images used for training
[imdsTrain, imdsTest] = splitEachLabel(imds, p, 'randomized');

[imdsTrain, imdsVali] = splitEachLabel(imdsTrain, (9/10), 'randomized');
[imdsTest, imdsDiscard] = splitEachLabel(imdsTest, testImg, 'randomized');

augmenter = imageDataAugmenter('RandXTranslation', [-20 20], 'RandYTranslation', [-20 20]);

% audsTrain = augmentedImageDatastore([64 64 1], imdsTrain, 'DataAugmentation', augmenter);
audsTrain = augmentedImageDatastore([64 64 1], imdsTrain);
audsVali = augmentedImageDatastore([64 64 1], imdsVali);
audsTest = augmentedImageDatastore([64 64 1], imdsTest);

% Preview augmented data
minibatch = preview(audsTrain);
figure; imshow(imtile(minibatch.input));

minibatch_vali = preview(audsVali);
figure; imshow(imtile(minibatch_vali.input));

% minibatch_test = preview(audsTest);
% figure; imshow(imtile(minibatch_test.input));
% pause();

%% CNN Architecture
% 15 layers
fprintf('Creating Network\n');
layers = [
    imageInputLayer([64 64 1]); % input layer - images are 64x64 grayscale
    convolution2dLayer([3 3], 8, 'Stride', 1, 'Padding', 'same');

    batchNormalizationLayer();
    reluLayer();
    maxPooling2dLayer([2 2], 'Stride', 2);
    convolution2dLayer([3 3], 16, 'Stride', 1, 'Padding', 'same', 'NumChannels', 8);

    batchNormalizationLayer();
    reluLayer();
    maxPooling2dLayer([2 2], 'Stride', 2);
    convolution2dLayer([3 3], 32, 'Stride', 1, 'Padding', 'same', 'NumChannels', 16);

    batchNormalizationLayer();
    reluLayer();
    fullyConnectedLayer(totalBeams);
    softmaxLayer();
    classificationLayer()
];

options = trainingOptions('sgdm', ...
    'Plots', 'training-progress', ...
    'Momentum', 0.5, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', audsVali, ...
    'ValidationPatience', 20, ...
    'MiniBatchSize', 64, ...
    'MaxEpochs', 20000, ...
    'InitialLearnRate', 0.01, ...
    'ExecutionEnvironment', 'gpu');

%% Train Network
beamNet = trainNetwork(audsTrain, layers, options); % LG beams

%% Test Network
[pred, scores] = classify(beamNet, audsTest);

%% Results
totalTest = testImg*totalBeams;
acc = nnz(imdsTest.Labels == pred) / totalTest;

```

```

figure; cm = confusionchart(imdsTest.Labels, pred);
title('Confusion Matrix')
cm.NormalizedValues
fprintf('Accuracy is %2.2f%%\n', acc*100);

misClassed = 1:1:255;
for i = 1:totalBeams
    misClassed(i) = testImg - cm.NormalizedValues(i,i);
end
figure; hold on
plot(misClassed, '^r'); grid on
axis([0 totalBeams 0 max(misClassed)+5]);
title('Inaccuracy Graph'); xlabel('Alphabet Beam Number'); ylabel('Number of Misclassifications');

```

Section 7: Calc_Sl.m – script to use images of Gaussian beam to estimate Cn2

```

%% Calculate Scintillation Index
% finds the pixel by pixel mean intensity and scintillation index of a
% given set of images
% Cn2 calculation from "Experimental study of electromagnetic Bessel-Gaussian Schell Model beams
% propagating in a turbulent channel"
clearvars; %close all; clc

%% Parameters
experimentFolder = sprintf('1024_Set'); % folder for experiment (organizes everything in matlab)
dataFolderC = sprintf('Zeta'); % this is the big folder everything goes in
idxImg1 = 2918; % starting image number 409 for weak
numImgs = 1000; % number of images collected
numRows = 64; % pixels in y
numCols = 64; % pixels in x
dt = 10; % number of frames per time step
row1 = 33; col1 = 30; spAvg = 4;

beamTotal = 512; % include Gaussian in this
cycleFrames = 11; % number of frames per trigger of camera (post cycling)
numLoops = 100; % number of loops from experiment
cycleRate = 20; % [Hz] - cycle rate of function generator

wvl = 633e-9; % [m] - wavelength
Z = 2.5; % [m] - propagation distance
w0 = 1e-3; % [m] - beam waist radius

%% Read in data
gaussFolder = sprintf('Beam_0000_2nd');

cd(experimentFolder);
cd(dataFolderC);
cd(gaussFolder);

beamImg = zeros(numRows,numCols,numImgs);

k = 1;
for l = 1:100
    m_start = idxImg1;
    m_end = idxImg1+9;
    for m = m_start:m_end
        imgFileName = sprintf('img%07d.png',m);
        beamImg(:, :, k) = imread(imgFileName);
        beamImg(:, :, k) = im2double(beamImg(:, :, k)); % convert to uint16 (just like camera)
        idxImg1 = idxImg1+1;
        k = k+1;
    end
    idxImg1 = idxImg1 + ((beamTotal-1)*cycleFrames)+1;
end
cd ../../..
fprintf('Downloaded Imgs\n');

```

```

%% %% Normalize Image Intensity values
% top = max(max(beamImg(:,:,:)));
% for i = 1:numRows
%     for j = 1:numCols
%         for k = 1:numImgs
%             beamImg(i,j,k) = beamImg(i,j,k)/top(1,1,k);
%         end
%     end
% end

%% Calculate the mean (pixel by pixel)
mu = zeros(numRows,numCols);
mu_ovrT = zeros(numRows, numCols, numImgs/dt);

for i = 1:numRows
    for j = 1:numCols
        sum = 0;
        for k = 1:numImgs
            sum = beamImg(i,j,k) + sum;
        end
        mu(i,j) = sum / numImgs;
    end
end
% over time
idx = 1;
for l = 1:dt:numImgs-dt
    for i = 1:numRows
        for j = 1:numCols
            sum = 0;
            for k = 1:(l+dt-1)
                sum = beamImg(i,j,k) + sum;
            end
            mu_ovrT(i,j,idx) = sum / dt;
        end
    end
    idx = idx+1;
end

fprintf('Mean\n');

%% Calculate SI (pixel by pixel)
SI = zeros(numRows, numCols);
SI_ovrT = zeros(numRows, numCols, numImgs/dt);

% all
for i = 1:numRows
    for j = 1:numCols
        sumVar = 0;
        for k = 1:numImgs
%             sumVar = (beamImg(i,j,k)-mu(i,j))^2 + sumVar;
            sumVar = (beamImg(i,j,k))^2 + sumVar;
        end
        sumVar = sumVar / numImgs;
        SI(i,j) = (sumVar / mu(i,j)^2) - 1;
    end
end
% over time
idx = 1;
for l = 1:dt:numImgs-dt
    for i = 1:numRows
        for j = 1:numCols
            sumVar = 0;
            for k = 1:(l+dt-1)
                sumVar = (beamImg(i,j,k))^2 + sumVar;
            end
            sumVar = sumVar / numImgs;
            SI_ovrT(i,j,idx) = (sumVar / mu_ovrT(i,j,idx)^2) - 1;
        end
    end
end

```

```

    idx = idx+1;
end

fprintf('SI\n');

%% Display
figure; imagesc(mu); colorbar
    title('Mean Intensity');

figure; imagesc(SI); colorbar
    title('Scintillation Index');

%% Determine Cn2 from SI
% Parameters
% ===== %
k = (2*pi)/wvl; % [1/m] - wave number

Lambda_0 = (2*Z) / (k*w0^2); % input plane diffraction parameter
Theta_0 = 1;

Lambda = Lambda_0 / (Theta_0^2 + Lambda_0^2);
Theta = Theta_0 / (Theta_0^2 + Lambda_0^2);
% ===== %

% Calculate Rytov Variance
rytov = zeros(numRows, numCols);
rytov_ovrT = zeros(numRows, numCols, numImgs/dt);

for i = 1:numRows
    for j = 1:numCols
        rytov(i,j) = SI(i,j) / (3.86*(0.40*((1+2*Theta)^2 + 4*Lambda^2)^(5/12) *
            cos((5/6)*atan((1+2*Theta)/2*Lambda))...
            - (11/16)*Lambda^(5/6)));
    end
end
% over time
for l = 1:(numImgs/dt)
    for i = 1:numRows
        for j = 1:numCols
            rytov_ovrT(i,j,l) = SI_ovrT(i,j,l) / (3.86*(0.40*((1+2*Theta)^2 + 4*Lambda^2)^(5/12) *
                cos((5/6)*atan((1+2*Theta)/2*Lambda))...
                - (11/16)*Lambda^(5/6)));
        end
    end
end

%% Calculate Cn2 at each pixel
Cn2 = zeros(numRows, numCols);
Cn2_ovrT = zeros(numRows, numCols, numImgs/dt);

for i = 1:numRows
    for j = 1:numCols
        Cn2(i,j) = rytov(i,j) / (1.23*k^(7/6)*Z^(11/6));
    end
end
% over time
for l = 1:(numImgs/dt)
    for i = 1:numRows
        for j = 1:numCols
            Cn2_ovrT(i,j,l) = rytov_ovrT(i,j,l) / (1.23*k^(7/6)*Z^(11/6));
        end
    end
end

%% avg Cn2
croppedCn2 = Cn2(row1:row1+spAvg, col1:col1+spAvg);
croppedCn2_ovrT = Cn2_ovrT(row1:row1+spAvg, col1:col1+spAvg, 1:numImgs/dt);

[rowCn2, colCn2] = size(croppedCn2);

```

```

% Sum together
sumCn2 = 0;
for i = 1:rowCn2
    for j = 1:colCn2
        sumCn2 = croppedCn2(i,j) + sumCn2;
    end
end
totalVals = rowCn2*colCn2;
avgCn2 = sumCn2 / totalVals;
% over time
for l = 1:(numImgs/dt)
    sumCn2 = 0;
    for i = 1:rowCn2
        for j = 1:colCn2
            sumCn2 = croppedCn2_ovrT(i,j,l) + sumCn2;
        end
    end
    totalVals = rowCn2*colCn2;
    avgCn2_ovrT(l) = sumCn2 / totalVals;
end

fprintf('The Average Cn2 for the center %d x %d pixels is %.4e\n', rowCn2, colCn2, avgCn2);

%% Check avg values
% C
check = 0;
for i = 1:length(avgCn2_ovrT)-1
    check = avgCn2_ovrT(i) + check;
end
check = check / (length(avgCn2_ovrT)-1)

%% Plot
T = 1/cycleRate; % s
totalTime = beamTotal*T*numLoops;

frames = dt:dt:numImgs-dt;
times = (totalTime/numLoops):(totalTime/numLoops):totalTime-(totalTime/numLoops);
figure; hold on;
    title('Cn2 vs Time'); ylabel('Cn2 [m^{-2/3}]'); xlabel('Time (s)')
    plot(times, avgCn2_ovrT(1:end-1), '^'); grid on

```

Section 8: *Nikishov_Sim.m* – script to run Nikishov simulation

```

%% Nikishov Random Phase Screen Split-Step Optical Wave Simulation
clearvars; % close all; clc
% Purpose: simulates propagation of optical wave through turbulent media,
% using Nikishov PSD for oceanic turbulence
% Produces TIFF file and AVI movie file of turbulent images

% Function Library:
% ang_spec_multi_prop.m - function to propagate beam in vacuum from Schmidt
% ft_sh_nikishovPS.m - generates Nikishov RPS for specified oceanic parameters
% ft_nikishovPS.m - generates high frequency PS from FT method
% ft2.m - fourier transform function from Schmidt
% genLG.m - generates complex amplitude of desired LG beam
% ift2.m - inverse fourier transform function from Schmidt
% LaguerreGen.m - generates Laguerre polynomial
% LGBeamCombinations.m - forms combined beams
% sh_nikishovPS.m - generates low frequency PS by augmenting FT with subharmonics (SH)

% INSTRUCTIONS:
% the only variables that should need to be changed are given in the File
% Naming and Optical Sim Parameters sections. The code has a lot of things
% commented out / parameters it calculates that honestly can be ignored, I
% just haven't gone through to clean it up yet! The only interesting ones
% to look at are the rytov number, which should stay low, and Cn2.
tic

%% File Naming
% saves TIFF image file and AVI movie file to folder given by "dataFolder"
% ===== THIS IS WHAT YOU CHANGE ===== %
dataFolder = sprintf('Test'); % this is the big folder everything goes in
dataFileName = sprintf('U03.tif'); % this is the name of the tiff file
videoFileName = sprintf('U03.avi'); % this is the name of the video file
% ===== %
simFolder = sprintf('Nikishov_Sim'); % folder to put all data in

%% Optical Simulation System Parameters
alpha_ord = [0 1 0 1]; % parameters for baseline "ALPHA" set of beams
alpha_top = [1 4 -6 9];
sigma_ord = [0 1 2 3 0 1 2 3];
sigma_top = [1 5 -9 2 -4 8 5 7];

% ===== THIS IS WHAT YOU CHANGE ===== %
N = 2^10; % number of grid points per side
D1 = 0.1; % [m] - side length of source plane
D2 = 0.1; % [m] - side length of observation plane
delta_1 = D1/N; % [m] - grid spacing at source
delta_n = D2/N; % [m] - grid spacing at receiver
Z = 100; % [m] - total propagation distance
wv1 = 633e-9; % [m] - wavelength
w0 = 1e-2; % [m] - beam waist radius
ord = 0; % order of Laguerre polynomial of LG beams - can be given as an array of basis beams
(combinations will be made automatically)
top = -6; % topological charge of transmitted LG beams
nscr = 10; % desired number of screens
numFrames = 1; % number of image frames to produce
cycleFrames = 1; % frames prior to cycling beam
c = 2; % adjustable parameter indicating sensitivity of model to turb (c = 2 captures ~97%, c = 4 - 99%) 2
< c < 8
eps = 1e-6; % [m^2 s^-3] - rate of dissipation of kinetic energy - typical values range from 1e-2
(surface) to 1e-8 (deep)
chi_T = 1e-7; % [K^2 s^-1] - rate of dissipation of mean-squared temperature and salinity - typical values
range from 1e-4 (surf) to 1e-10 (deep)
omega = -5; % scale factor ranging from -5 (temperature only) to 0 (salinity only) - not currently being
used bc I am using simplified temp only spectrum
% ===== %

```

```

L1 = delta_1*N; % [m] - side length at source
L2 = delta_n*N; % [m] - side length at receiver
k = (2*pi) / wvl; % [rad/m] - wavenumber
[x1, y1] = meshgrid((-N/2 : N/2-1)*delta_1); % grid size: L1 x L1 [m]
[theta1, r1] = cart2pol(x1, y1);

% Generate all possible combinations of n base beams
n_beam = length(top);
beamTotal = 2^n_beam - 1;
LG = zeros(N,N,n_beam);
for i = 1:n_beam
    bin(:, :, i) = zeros([1 n_beam]);
end
for j = 1:n_beam
    LG(:, :, j) = genLG(top(j), ord(j), wvl, x1, y1, w0); % topological charge, order of LG polynomial,
    grid, beam waist
end
[U, beam, beamBin] = LGBeamCombinations(n_beam, beamTotal, LG, bin); % generates array of complex
amplitude of all input beams

% % plot beam at source
for s = 1:beamTotal
    Uin = U(:, :, s); % plot first beam
    figure; imagesc(abs(Uin)); % colormap(gray);
    title('Beam Amplitude at Source'); xlabel([num2str(L1) ' [m]']); ylabel([num2str(L1) ' [m]']);
    figure; imagesc(angle(Uin));
    title('Beam Phase at Source'); xlabel([num2str(L1) ' [m]']); ylabel([num2str(L1) ' [m]']);
end

%% Oceanic Parameters
C0 = 0.72; % constant
C1 = 2.35; % constant
Pr_T = 7; % typical Prandtl number for temperature spectrum
Pr_S = 700; % typical Prandtl number for salinity spectrum

A_T = (C0*(C1^2)) / Pr_T; % temperature constant
A_S = (C0*(C1^2)) / Pr_S; % salinity constant - unused bc only want temp
A_TS = 0.5*C0*(C1^2)*(1/Pr_T + 1/Pr_S);
% omega = -5; % for temp only, we take omega as 5 or -inf. for salinity only, omega = 0

eps_surf = 1e-2; % [m^2 s^-3] - typical value from Korotkova
eps_deep = 1e-8; % [m^2 s^-3] - typical value from Korotkova
chi_T_surf = 1e-4; % [K^2 s^-1] - typical value from Korotkova
chi_T_deep = 1e-10; % [K^2 s^-1] - typical value from Korotkov
eps_kor = 1e-5; % from Korotkova paper
chi_T_kor = 1e-7; % from Korotkova paper
alpha_T = 2.6e-4; % L/degree
v_w = 1e-6; % [m^2/s] kinematic viscosity of water at 20 deg Celcius

% % ===== THIS IS WHAT YOU CHANGE ===== %
% eps = 1e-6; % rate of dissipation of kinetic energy
% chi_T = 1e-7; % rate of dissipation of mean-squared temperature and salinity
% omega = -5; % scale factor ranging from -5 (temperature only) to 0 (salinity only)
% % ===== %

l0 = 7.4*(v_w^3 / eps)^(1/4);
k_eta = k*l0; %kn - Kolmogorov inner-scale number

% Check Cn2 value
Cn2 = (alpha_T^2)*(eps^(-1/3))*chi_T % from Korotkova

%% Plot Nikishov Spectra
k_eta_norm = logspace(-3, 3, 1000);

w_val = [0 -0.2 -0.5 -1 -5]; % rate of temp and salinity contributions to OT (-5 temp to 0 sal)

delta = 1.5*(C1^2).*((k_eta_norm).^(4/3)) + (C1^3).*(k_eta_norm.^2);

for i = 1:length(w_val)

```

```

w = w_val(i);

norm_Nikishov_PS(i, :) = C0.*(1 + C1.*((k_eta_norm).^(2/3))).*...
    (((w^2).*exp(-A_T.*delta)) + exp(-A_S.*delta) - (2*w.*exp(-A_TS.*delta)))/ (w^2 + 1 - 2*w));
end

% Plot range
% figure;
% for i = 1:5
%     semilogx(k_eta_norm, norm_Nikishov_PS(i, :)); hold on
% end
% title('Nikishov Power Spectrum from Korotkova');
% xlabel('k\eta'); ylabel('Normalized Power Spectra');
% legend('w = 0', 'w = -0.2', 'w = -0.5', 'w = -1', 'w = -5');
% axis([0.001 1000 0 8]);

% Just Temperature and Salinity Normalized Spectra from Korotkova

norm_phi_T = C0.*(1 + C1.*((k_eta_norm).^(2/3))).*exp(-A_T.*delta);
norm_phi_S = C0.*(1 + C1.*((k_eta_norm).^(2/3))).*exp(-A_S.*delta);

% Plot just full temp and full salinity (w = 0, w = -inf)
% figure;
% semilogx(k_eta_norm, norm_phi_T, 'b-'); hold on
% semilogx(k_eta_norm, norm_phi_S, 'r-');
% title('Temperature and Salinity Spectra in Turbulent Ocean from Korotkova');
% xlabel('k\eta'); ylabel('Normalized Power Spectra');
% legend('Temperature Spectrum', 'Salinity Spectrum');
% axis([0.001 1000 0 8]);

%% Validate Screen Properties
% these equations are from Lu 2014
rho0_pw = (3.603e-7*k^2*Z*eps^(-1/3)*(chi_T/(2*(omega^2)))*(1.116*omega^2 - 2.235*omega + 1.119))^(-3/5);
% [m] - radius of coherence in oceanic turb for plane wave
rho0_sw = (3.603e-7*k^2*Z*eps^(-1/3)*(chi_T/(2*(omega^2)))*(0.419*omega^2 - 0.838*omega + 0.419))^(-3/5);
% [m] - radius of coherence in oceanic turb for spherical wave

r0_pw = 2.1*rho0_pw; % [m] - from Toselli paper for condition rho0 >> l0
r0_sw = 2.1*rho0_sw % [m] - coherence diameter of speherical wave should be ~10-20 cm
p = linspace(0, Z, 1e3);

% log-amplitude variance of plane wave (should be rytov < 0.25)
rytov = 0.563 * k^(7/6) * sum(Cn2 * (1-p/Z).^(5/6) .* p.^(5/6) * (p(2)-p(1)))

%% Check Turbulent Prop Constraints
D1p = D1 + (c*vw1*Z)/r0_sw;
D2p = D2 + (c*vw1*Z)/r0_sw;

% constraint 1 - need smaller number
delta_n_max = ((vw1*Z)-(D2p*delta_1)) / D1p % [m] - maximum grid spacing at observation plane

% constraint 2 - need number bigger than
N_min = (D1p/(2*delta_1)) + (D2p/(2*delta_n)) + ((vw1*Z)/(2*delta_1*delta_n)) % [m] - minimum number of
samples per side of grid

% constraint 4
z_step_max = ((min(delta_1,delta_n))^2 * N) / vw1 % maximum step size between partial propagations
nscr_min = ceil(Z/z_step_max) + 1

%% Test Propagation in Vacuum
n = nscr; % number of partial propagations
z_step_size = Z/n; % [m]
z = (1:n-1) * Z / n-1;

sg = exp(-(x1/(0.47*N*delta_1)).^16) .* exp(-(y1/(0.47*N*delta_1)).^16); % super-Gaussian function
% t = repmat(sg, [1 1 n]);

% propagate beam
% Uin = U(:, :, 1); % use just first beam

```

```

% [x2, y2, Uout_vac] = ang_spec_multi_prop(Uin, wvl, delta_1, delta_n, z, t); % function from Schmidt for
beam prop
%
% % plot output
% figure; imagesc(abs(Uout_vac)); % colormap(gray);
% title('Beam Ampitude at Receiver (Vacuum)'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) '
[m]']);
% figure; imagesc(angle(Uout_vac));
% title('Beam Phase at Receiver (Vacuum)'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) '
[m]']);

% visually verify output using predicted Gaussian beam radius
zR = (pi*w0^2) / wvl; % [m] - Rayleigh range
w_pred = w0*(1 + (Z/zR)^2)^(1/2) % [m] - predicted beam radius at receiver

%% Turbulent Propagation
zt = [0 z]; % phase screen locations
delta_z = zt(2:n) - zt(1:n-1); % distances between screens

alpha = zt / zt(n);
delta_sim = (1-alpha)*delta_1 + alpha*delta_n;
sg = repmat(sg, [1 1 n]);

% simulate propagation over time
Uout_total = zeros(N,N,numFrames);
idxFrame = 1;
while idxFrame <= numFrames
    for jj = 1:beamTotal
        Uin = U(:, :, jj);
        for ii = 1:cycleFrames
            % ===== THIS CREATES ONE FRAME ===== %
            % Generate Phase Screens
            phi = zeros(N,N,nscr);
            for i = 1:nscr
                [phi(:, :, i), phi_hi(:, :, i), phi_lo(:, :, i)] = ft_sh_nikishovPS(N, delta_1, omega, eps,
chi_T); % function which uses oceanic parameters and Nikishov PSD to generate RPS
            % figure; imagesc(abs(phi(:, :, i))); title('Nikishov Phase Screen')
            end
            % % plot phase screen
            % figure; imagesc(abs(phi(:, :, 1)));
            % title('Sample Nikishov Phase Screen'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2)
' [m]']);

            [xn, yn, Uout] = ang_spec_multi_prop(Uin, wvl, delta_1, delta_n, z, sg.*exp(1i*phi)); %
propagation function from Schmidt
            Uout_total(:, :, idxFrame) = Uout;

            % capture image
            figure; imagesc(abs(Uout)); colormap(gray);
            set(gca, 'position', [0 0 1 1], 'units', 'normalized');
            F = getframe(gcf);
            [Ir(:, :, :, idxFrame), Map] = frame2im(F);
            fprintf('Frame %d\n', idxFrame);
            close

            % plot output
            figure; imagesc(abs(Uout)); % colormap(gray);
            title('Beam Ampitude at Receiver'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) '
[m]']);
            figure; imagesc(angle(Uout));
            title('Beam Phase at Receiver'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) '
[m]']);

            % [xn, yn, Uout_hi] = ang_spec_multi_prop(Uin, wvl, delta_1, delta_n, z, sg.*exp(1i*phi_hi));
% propagation function from Schmidt
% % plot output
% figure; imagesc(abs(Uout_hi)); % colormap(gray);
% title('Beam Ampitude at Receiver from FT'); xlabel([num2str(L2) ' [m]']);
ylabel([num2str(L2) ' [m]']);

```

```

%           figure; imagesc(angle(Uout_hi));
%           title('Beam Phase at Receiver from FT'); xlabel([num2str(L2) ' [m]']);
ylabel([num2str(L2) ' [m]']);
%
%           [xn, yn, Uout_lo] = ang_spec_multi_prop(Uin, wvl, delta_1, delta_n, z, sg.*exp(1i*phi_lo));
% propagation function from Schmidt
%           % plot output
%           figure; imagesc(abs(Uout_lo)); % colormap(gray);
%           title('Beam Ampitude at Receiver from SH'); xlabel([num2str(L2) ' [m]']);
ylabel([num2str(L2) ' [m]']);
%           figure; imagesc(angle(Uout_lo));
%           title('Beam Phase at Receiver from SH'); xlabel([num2str(L2) ' [m]']);
ylabel([num2str(L2) ' [m]']);
% ===== %
    idxFrame = idxFrame + 1;
end
end
end
% plot 1 phase screen
% figure; imagesc(abs(phi(:,:,1))); colorbar; % colormap(gray)
% title('Nikishov FT SH Phase Screen'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) ' [m]']);
%
% figure; imagesc(abs(phi_hi(:,:,1))); colorbar; % colormap(gray)
% title('Nikishov FT Phase Screen'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) ' [m]']);
%
% figure; imagesc(abs(phi_lo(:,:,1))); colorbar; % colormap(gray)
% title('Nikishov SH Phase Screen'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) ' [m]']);

%% Save images and video
cd(simFolder);
mkdir(dataFolder);
cd(dataFolder);

fprintf('Tiff Save Start\n')
imwrite(Ir(:,:,1), dataFileName);
for ii = 2:idxFrame-1
    imwrite(Ir(:,:,ii), dataFileName, 'WriteMode', 'append');
    fprintf('Tiff Frame %d\n', ii);
end
fprintf('Tiff Save End\n');

fprintf('Video Save Start\n');
vid = VideoWriter(videoFileName, 'Uncompressed AVI');
open(vid);
for vv = 1:idxFrame-1
    writeVideo(vid, Ir(:,:,vv));
end
close(vid)
fprintf('Video Save End\n');

cd ../../

elapsed_minutes_sim = floor(toc/60);
elapsed_seconds_sim = round((toc/60) - floor(toc/60))*60, 0);
fprintf('Total time to run sim: %d:%d\n', elapsed_minutes_sim, elapsed_seconds_sim)

```

Section 9: genSLMTurbScrns.m – script to generate and save SLM turbulence screens for hybrid turbulence

```

%% Generate Nikishov Screens for SLM

%% Generate Screens for Nikishov Propagation

% clearvars; close all; clc
% Function Library:
% ft_sh_nikishovPS.m - generates Nikishov screen
% ft_nikishovPS.m
% sh_nikishovPS.m

%% Simulation Parameters
% ===== THIS IS WHAT YOU CHANGE ===== %
phaseScrnFolder = sprintf('eps_-5_chiT_-7_20'); % folder to put all screen files in
N = 2^8; % number of grid points - set to 256
Z = 30; % [m] - total propagation distance
wvl = 633e-9; % [m] - wavelength
nscr = 20; % desired number of screens per frame
eps = 1e-5; % [m^2 s^-3] - rate of dissipation of kinetic energy - typical values range from 1e-2
(surface) to 1e-8 (deep)
chi_T = 1e-7; % [K^2 s^-1] - rate of dissipation of mean-squared temperature and salinity - typical values
range from 1e-4 (surf) to 1e-10 (deep)
% ===== %

% ===== DONT CHANGE ===== %
D1 = 0.10; % [m] - size of source plane
D2 = 0.10; % [m] - size of observation plane
delta_1 = D1/N; % [m] - grid spacing at source
delta_n = D2/N; % [m] - grid spacing at receiver
omega = -5;

k_wv = (2*pi) / wvl; % [rad/m] - wavenumber
[x1, y1] = meshgrid((-N/2 : N/2-1)*delta_1); % grid size: L1 x L1 [m]
[theta1, r1] = cart2pol(x1, y1);

%% Oceanic Parameters
C0 = 0.72; % constant
C1 = 2.35; % constant
Pr_T = 7; % typical Prandtl number for temperature spectrum
Pr_S = 700; % typical Prandtl number for salinity spectrum

A_T = (C0*(C1^2)) / Pr_T; % temperature constant
A_S = (C0*(C1^2)) / Pr_S; % salinity constant - unused bc only want temp
A_TS = 0.5*C0*(C1^2)*(1/Pr_T + 1/Pr_S);
% omega = -5; % for temp only, we take omega as 5 or -inf. for salinity only, omega = 0

eps_surf = 1e-2; % [m^2 s^-3] - typical value from Korotkova
eps_deep = 1e-8; % [m^2 s^-3] - typical value from Korotkova
chi_T_surf = 1e-4; % [K^2 s^-1] - typical value from Korotkova
chi_T_deep = 1e-10; % [K^2 s^-1] - typical value from Korotkov
eps_kor = 1e-5; % from Korotkova paper
chi_T_kor = 1e-7; % from Korotkova paper
alpha_T = 2.6e-4; % L/degree
v_w = 1e-6; % [m^2/s] kinematic viscosity of water at 20 deg Celcius

% % ===== THIS IS WHAT YOU CHANGE ===== %
% eps = 1e-6; % rate of dissipation of kinetic energy
% chi_T = 1e-7; % rate of dissipation of mean-squared temperature and salinity
% omega = -5; % scale factor ranging from -5 (temperature only) to 0 (salinity only)
% % ===== %

% Check Cn2 value
Cn2 = (alpha_T^2)*(eps^(-1/3))*chi_T % from Korotkova

%% Validate Screen Properties
% these equations are from Lu 2014

```

```

rho0_pw = (3.603e-7*k_wv^2*Z*eps^(-1/3)*(chi_T/(2*(omega^2)))*(1.116*omega^2 - 2.235*omega + 1.119))^(-
3/5); % [m] - radius of coherence in oceanic turb for plane wave
rho0_sw = (3.603e-7*k_wv^2*Z*eps^(-1/3)*(chi_T/(2*(omega^2)))*(0.419*omega^2 - 0.838*omega + 0.419))^(-
3/5); % [m] - radius of coherence in oceanic turb for spherical wave

r0_pw = 2.1*rho0_pw; % [m] - from Toselli paper for condition rho0 >> l0
r0_sw = 2.1*rho0_sw % [m] - coherence diameter of speherical wave should be ~10-20 cm
p = linspace(0, Z, 1e3);

% log-amplitude variance of plane wave (should be rytov < 0.25)
rytov = 0.563 * k_wv^(7/6) * sum(Cn2 * (1-p/Z).^(5/6) .* p.^(5/6) * (p(2)-p(1)))

%% Generate Phase Screens
mkdir(phaseScrnFolder);

phi = zeros(N,N,nscr);
for i = 1:nscr
    [phi(:, :, i), phi_hi(:, :, i), phi_lo(:, :, i)] = ft_sh_nikishovPS(N, delta_1, omega, eps, chi_T); %
function which uses oceanic parameters and Nikishov PSD to generate RPS
    figure; imagesc(abs(phi(:, :, i))); title('Nikishov Phase Screen')
    SLM_Phase = phi(:, :, i) + pi;

    SLM_Phase = SLM_Phase/max(max(SLM_Phase))*255;
    figure; imagesc(abs(SLM_Phase))

    cd(phaseScrnFolder);
    scrnName = sprintf('%03d.bmp', i);
    imwrite(uint8(SLM_Phase), scrnName, 'bmp');
end

```

Section 10: *ft_nikishovPS.m* – function to make FT phase screen

```

function phz = ft_nikishovPS(N, delta_in, omega, eps, chi_T)

% test parameters
% N = 2^11;
% D = 0.1;
% delta_in = D/N;
% eps = 1e-5;
% chi_T = 1e-7;

% Preliminary Grid Set-up and Constants
D = N*delta_in;

a = pi/delta_in; % [2*pi*(N/m)] - grid points in frequency domain
kx = (-a:2*a/N:a-2*a/N)'; % frequency domain indexing
ky = kx;
[Kx, Ky] = meshgrid(kx, ky);

% Phase screen parameters
k = sqrt(Kx.^2 + Ky.^2); % spatial wave vector

C0 = 0.72; % constant
C1 = 2.35; % constant
Pr_T = 7; % typical Prandtl number for temperature spectrum
Pr_S = 700; % typical Prandtl number for salinity spectrum

A_T = (C0*(C1^(-2))) / Pr_T; % temperature constant
A_S = (C0*(C1^(-2))) / Pr_S; % salinity constant - unused bc only want temp
A_TS = 0.5*C0*(C1^(-2))*(1/Pr_T + 1/Pr_S);

alpha_T = 2.6e-4; % L/degree
v_w = 1e-6; % [m^2/s] kinematic viscosity of water at 20 deg Celcius

l0 = 7.4*(v_w^3 / eps)^(1/4);
k_eta = k*l0; %kn - factors in Kolmogorov inner-scale number

```

```

% delta = 1.5*(C1^2).*((k_eta).^(4/3)) + (C1^3).*(k_eta.^2);
del_f = 1/(N*delta_in); % frequency grid spacing
delta_s = 1.5*(C1^2).*((k_eta).^(4/3)) + (C1^3).*(k_eta.^2);

Cn2 = (alpha_T^2)*(eps^(-1/3))*chi_T; % from Korotkova
Cn2_2 = (1e-8)*(eps^(-1/3))*chi_T; % from Pan

check1 = C0.*(alpha_T^2)*(eps^(-1/3))*chi_T;
check2 = 0.388*(1e-8)*(eps^(-1/3))*chi_T;

% Generate screen
phi = C0.*(alpha_T^2)*(eps^(-1/3))*chi_T.*(k^(-5/3)).*(1 + C1.*((k_eta).^(2/3))).*exp(-A_T*delta_s);
phi(N/2+1, N/2+1) = 0;

% figure; imagesc(abs(Niki_phi)); colorbar
% figure; imagesc(angle(Niki_phi)); colorbar

cn_2 = (randn(N) + 1i*randn(N)) .* sqrt(phi) * (2*pi*del_f); % adds random Gaussian coefficients - screens
shouldn't correlate

phz = real(ift2(cn_2, 1));
phz = phz./max(max(phz));

% figure; imagesc(abs(phz)); title('Nikishov FT Phase Screen'); colorbar

```

Section 11: sh_nikishovPS.m – function to make SH phase screen

```
function phz = sh_nikishovPS(N, delta_in, omega, eps, chi_T)
% test parameters
% N = 2^11;
% D = 0.1;
% delta_in = D/N;
% eps = 1e-5;
% chi_T = 1e-7;

% generate high and low frequency screen - ft_sh_phase_screen(r0, N, delta, L0, l0)
D = N*delta_in;

[x, y] = meshgrid((-N/2 : N/2-1) * delta_in);

C0 = 0.72; % constant
C1 = 2.35; % constant
Pr_T = 7; % typical Prandtl number for temperature spectrum
Pr_S = 700; % typical Prandtl number for salinity spectrum

A_T = (C0*(C1^2)) / Pr_T; % temperature constant
A_S = (C0*(C1^2)) / Pr_S; % salinity constant - unused bc only want temp
A_TS = 0.5*C0*(C1^2)*(1/Pr_T + 1/Pr_S);

alpha_T = 2.6e-4; % L/degree
v_w = 1e-6; % [m^2/s] kinematic viscosity of water at 20 deg Celcius

l0 = 7.4*(v_w^3 / eps)^(1/4);

phz_lo = zeros(N,N);
% loop over frequency grids with the spacing 1/(3^p*L)
for p = 1:3
    % setup PSD
    del_f = 1/ (3^p*D); % frequency grid spacing [1/m]
    fx = (-1 : 1) * del_f;

    [fx, fy] = meshgrid(fx);
    [th, f] = cart2pol(fx, fy);
    k_eta = f*l0;
    delta_s = 1.5*(C1^2).*((k_eta).^(4/3)) + (C1^3).*(k_eta.^2);

    phi = C0.*(alpha_T^2)*(eps^(-1/3))*chi_T.*(f^(-5/3)).*(1 + C1.*((k_eta).^(2/3))).*exp(-A_T*delta_s);
    phi(2,2) = 0;

    cn = (randn(3) + 1i*randn(3)) .* sqrt(phi)*del_f;

    SH = zeros (N);
    % loop over frequencies on this grid
    for ii = 1:3
        SH = SH + cn(ii) * exp(1i*2*pi*(fx(ii)*x+fy(ii)*y));
    end
    phz_lo = phz_lo + SH; % accumulate subharmonics
end
phz = real(phz_lo) - mean(real(phz_lo(:)));
phz = phz./max(max(phz));

% figure; imagesc(abs(phz)); title('Nikishov Spectrum SH Phase Screen'); colorbar
```

Section 12: ft_sh_nikishovPS.m – function to make combined phase screen

```
function [phz, phz_hi, phz_lo] = ft_sh_nikishovPS(N, delta_in, omega, eps, chi_T)

% test parameters
% eps_deep = 1e-8; % [m^2 s^-3] - typical value from Korotkova
% chi_T_deep = 1e-10;
%
% N = 2^10;
% D = 0.1;
% delta_in = D/N;
% omega = -5;
% eps = 1e-5;
% chi_T = 1e-7;

phz_hi = ft_nikishovPS(N, delta_in, omega, eps, chi_T);
phz_lo = sh_nikishovPS(N, delta_in, omega, eps, chi_T);

% figure; imagesc(abs(phz_hi)); title('Nikishov Spectrum FT Phase Screen'); colorbar
% figure; imagesc(abs(phz_lo)); title('Nikishov Spectrum SH Phase Screen'); colorbar

phz = phz_hi + phz_lo;

% figure; imagesc(abs(phz)); title('Nikishov Spectrum FT SH Phase Screen'); colorbar
```

Section 13: Screen_Vali.m – function to validate the screens are non-correlated and produced properly

```

%% Screen Statistics Validation
clearvars; close all; clc
% Function Library:
% nikishovPS.m - generates Nikishov screen

%% Simulation Parameters
% ===== THIS IS WHAT YOU CHANGE ===== %
N = 2^10; % number of grid points
D1 = 0.10; % [m] - size of source plane
D2 = 0.10; % [m] - size of observation plane
delta_1 = D1/N; % [m] - grid spacing at source
delta_n = D2/N; % [m] - grid spacing at receiver
Z = 10; % [m] - total propagation distance
wvl = 633e-9; % [m] - wavelength
nscr = 16; % desired number of screens
omega = -5; %
% ===== %

L1 = delta_1*N; % [m] - side length at source
L2 = delta_n*N; % [m] - side length at receiver
k_wv = (2*pi) / wvl; % [rad/m] - wavenumber
[x1, y1] = meshgrid((-N/2 : N/2-1)*delta_1); % grid size: L1 x L1 [m]
[theta1, r1] = cart2pol(x1, y1);

%% Oceanic Parameters
a = pi/delta_1; % [2*pi*(N/m)] - grid points in frequency domain
kx = (-a:2*a/N:a-2*a/N)'; % frequency domain indexing
ky = kx;
[Kx, Ky] = meshgrid(kx,ky);

k = sqrt(Kx.^2 + Ky.^2); % spatial wave vector

C0 = 0.72; % constant
C1 = 2.35; % constant
Pr_T = 7; % typical Prandtl number for temperature spectrum
Pr_S = 700; % typical Prandtl number for salinity spectrum

A_T = (C0*(C1^-2)) / Pr_T; % temperature constant
A_S = (C0*(C1^-2)) / Pr_S; % salinity constant - unused bc only want temp
A_TS = 0.5*C0*(C1^-2)*(1/Pr_T + 1/Pr_S);

eps_surf = 1e-2; % [m^2 s^-3] - typical value from Korotkova
eps_deep = 1e-8; % [m^2 s^-3] - typical value from Korotkova
chi_T_surf = 1e-4; % [K^2 s^-1] - typical value from Korotkova
chi_T_deep = 1e-10; % [K^2 s^-1] - typical value from Korotkova
eps_13 = 1e-5; % for Cn2 value at the order of -13
chi_T_13 = 1e-7; % for Cn2 value at the order of -13
alpha_T = 2.6e-4; % L/degree
v_w = 1e-6; % [m^2/s] kinematic viscosity of water at 20 deg Celcius

% ===== THIS IS WHAT YOU CHANGE ===== %
eps = 1e-5; % rate of dissipation of kinetic energy
chi_T = 1e-7; % rate of dissipation of mean-squared temperature and salinity
% ===== %

l0 = 7.4*(v_w^3 / eps)^(1/4);
k_eta = k*l0; %kn - Kolmogorov inner-scale number

% Check Cn2 value
Cn2 = (alpha_T^2)*(eps^(-1/3))*chi_T % from Korotkova

%% Plot Nikishov Spectra
k_eta_norm = logspace(-3, 3, 1000);

```

```

w_val = [0 -0.2 -0.5 -1 -5]; % rate of temp and salinity contributions to OT (-5 temp to 0 sal)

delta = 1.5*(C1^2).*((k_eta_norm).^(4/3)) + (C1^3).*(k_eta_norm.^2);

for i = 1:length(w_val)
    w = w_val(i);

    norm_Nikishov_PS(i, :) = C0.*(1 + C1.*((k_eta_norm).^(2/3))).*...
        (((w^2).*exp(-A_T.*delta)) + exp(-A_S.*delta) - (2*w.*exp(-A_TS.*delta)))/ (w^2 + 1 - 2*w));
end

% Plot range
figure;
for i = 1:5
    semilogx(k_eta_norm, norm_Nikishov_PS(i, :)); hold on
end
title('Nikishov Power Spectrum from Korotkova');
xlabel('k\eta'); ylabel('Normalized Power Spectra');
legend('w = 0', 'w = -0.2', 'w = -0.5', 'w = -1', 'w = -5');
axis([0.001 1000 0 8]);

% Just Temperature and Salinity Normalized Spectra from Korotkova

norm_phi_T = C0.*(1 + C1.*((k_eta_norm).^(2/3))).*exp(-A_T.*delta);
norm_phi_S = C0.*(1 + C1.*((k_eta_norm).^(2/3))).*exp(-A_S.*delta);

% Plot just full temp and full salinity (w = 0, w = -inf)
figure;
semilogx(k_eta_norm, norm_phi_T, 'b-'); hold on
semilogx(k_eta_norm, norm_phi_S, 'r-');
title('Temperature and Salinity Spectra in Turbulent Ocean from Korotkova');
xlabel('k\eta'); ylabel('Normalized Power Spectra');
legend('Temperature Spectrum', 'Salinity Spectrum');
axis([0.001 1000 0 8]);

%% Validate Screen Properties
% these equations are from Lu 2014
rho0_pw = (3.603e-7*k_wv^2*Z*eps^(-1/3)*(chi_T/(2*(omega^2)))*(1.116*omega^2 - 2.235*omega + 1.119))^(-3/5); % [m] - radius of coherence in oceanic turb for plane wave
rho0_sw = (3.603e-7*k_wv^2*Z*eps^(-1/3)*(chi_T/(2*(omega^2)))*(0.419*omega^2 - 0.838*omega + 0.419))^(-3/5); % [m] - radius of coherence in oceanic turb for spherical wave

r0_pw = 2.1*rho0_pw; % [m] - from Toselli paper for condition rho0 >> 10
r0_sw = 2.1*rho0_sw % [m] - coherence diameter of speherical wave should be ~10-20 cm
p = linspace(0, Z, 1e3);

% log-amplitude variance of plane wave (should be rytov < 0.25)
rytov = 0.563 * k_wv^(7/6) * sum(Cn2 * (1-p/Z).^(5/6) .* p.^(5/6) * (p(2)-p(1)))

%% Generate Phase Screens
phi = zeros(N,N,nscr);
for i = 1:nscr
    % phi(:, :, i) = nikishovPS(N, delta_1, omega, eps, chi_T); % function which uses oceanic parameters and Nikishov PSD to generate RPS
    % plot phase screen
    % figure; imagesc(abs(phi(:, :, i)));
    % title('Nikishov Spectrum Phase Screen'); xlabel([num2str(L2) ' [m]']); ylabel([num2str(L2) ' [m]']);
end

%% Generate screens for comparison
delta = 1.5*(C1^2).*((k_eta).^(4/3)) + (C1^3).*(k_eta.^2);
% Simplified temperature only Nikishov spectrum
phz = C0.*(alpha_T^2)*(eps^(-1/3))*chi_T.*(k^(-5/3)).*(1 + C1.*((k_eta).^(2/3))).*exp(-A_T.*delta);

figure; imagesc(abs(phz)); colorbar
figure; imagesc(angle(phz)); colorbar

```

```

ce = randn(size(phz)) + 1i*randn(size(phz)); % adds random Gaussian coefficients - screens shouldn't
correlate
% phi_s = real(fft2(fftshift(sqrt(phz).*ce)));
phi_s = real(fft2(ce,1));

figure; imagesc(abs(phi_s)); title('Spectrum from Nikshov w/out salinity'); colorbar
figure; imagesc(angle(phi_s)); title('Spectrum from Nikshov w/out salinity'); colorbar

phz = C0.*(alpha_T^2)*(eps^(-1/3))*chi_T.*(k^(-5/3)).*(1 + C1.*((k_eta).^(2/3))).*exp(-A_T*delta);
phz(N/2+1, N/2+1) = 0;

figure; imagesc(abs(phz)); colorbar
figure; imagesc(angle(phz)); colorbar

cn = (randn(N) + 1i*randn(N)) .* sqrt(phz)*a; % adds random Gaussian coefficients - screens shouldn't
correlate
phi_sch = real(fft2(cn, 1));

figure; imagesc(abs(phi_sch)); title('Spectrum from Nikshov w/out salinity'); colorbar
figure; imagesc(angle(phi_sch)); title('Spectrum from Nikshov w/out salinity'); colorbar

```

Neural Network Classification of Structured Light Through Optical Turbulence

W. A. Jarrett^a, S. Avramov-Zamurovic^a, C. Nelson^a, J. Esposito^a, M. W. Hyde^b

United States Naval Academy, 597 McNair Rd., Hopper Hall, Annapolis, Maryland USA 21402

Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, OH 45433

ABSTRACT

We consider a design of a high-speed wireless optical communication system that involves generating an alphabet using structured light by encoding the transmitted information using Laguerre-Gaussian (LG) beams carrying orbital angular momentum (OAM) [1]. LG beams carrying OAM are chosen for this system due to their resilience when propagated through complex environments. Following propagation through optical turbulence, light intensity patterns are decoded with high accuracy using a convolutional neural network (CNN), commonly used for image classification applications. CNNs only see and learn from the diversity of the received images, they do not recognize how the light intensity distribution has been generated. This motivates us to investigate how the CNN handles and reacts to different forms of structured light that have the same intensity pattern on reception and are transmitted through the same environmental conditions.

To compare the effects of structured light on the performance of a CNN we constructed an alphabet using LG beams carrying OAM [2] and 2D projected Far Field Images (FFI) [5]. Under the conditions where there is no induced optical turbulence, the received images have approximately the same intensity pattern, despite differing formation methods. Each form of the structured light is generated using a spatial light modulator (SLM). The slight differences in the generated intensity patterns are a result of the optical artifacts of the SLM and the method used to create the SLM phase screen. The primary comparison metric in this research is the classification accuracy of the CNN when individually trained on images of each type of light. This will provide insight into which form of light is potentially the most resilient to be utilized in receivers supported by CNN.

Our experimental set uses a 632.8 nm He-Ne laser and an SLM to generate structured light and propagate two forms of structured light in underwater optical turbulence over a ~ 2.5 m propagation path. Induced underwater optical turbulence is created using a heater that allows for the control and estimation of turbulence strength. The images are captured by a camera, and the CNN is comprised of 15 layers, reducing computational complexity [4]. The beams are propagated through strong optical turbulence, corresponding to C_n^2 values of approximately $10^{-11} \text{ m}^{-2/3}$. Based on preliminary testing, the CNN has been able to accurately learn the resulting variations for all forms of light on as little as 50 images. This result further strengthens the evidence for the resilience of machine learning-based communication systems in harsh environments.

Keywords: laser propagation, Laguerre-Gaussian beam, orbital angular momentum, optical turbulence

1. INTRODUCTION

Light carrying orbital angular momentum (OAM) has garnered recent interest in communication systems due to its ability to be multiplexed for a higher bandwidth. One communication system proposed utilized Laguerre-Gaussian (LG) beams carrying OAM, where information was spatially multiplexed into the resulting intensity patterns of the light [1]. The information was then decoded using a deep convolutional neural network (CNN). It was shown that CNNs are capable of high levels of classification accuracy even in the presence of significant optical turbulence [2] and attenuation due to scattering and absorption [3], the primary obstacles in underwater wireless optical communication systems. Figure 1 illustrates an example of a 15-image alphabet of LG beams, and provides both the theoretical and experimental intensity patterns to create context.

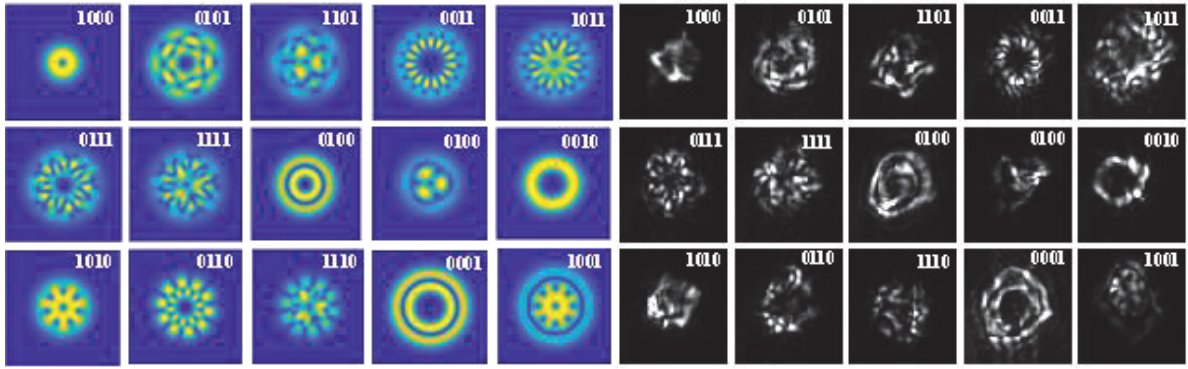


Figure 1: (a) theoretical LG beams and (b) experimental LG beams captured after propagation through underwater medium with thermally-generated optical turbulence

However, the CNN only uses the intensity pattern of the light at the receiver when classifying messages, it does not factor in how the light is formed and with what properties. This provided a motivation for further evaluation of neural network classification of light by way of a comparison between forms of structured light that carry highly diverse properties. For this research, LG beams carrying OAM are compared to 2D projected Far Field Images (FFIs). The FFIs were utilized because they lack spatial coherence, are uncollimated, and do not carry OAM.

2. IMAGE GENERATION

Both types of structured light were generated using a 256 x 256 pixel phase-only spatial light modulator (SLM). The screens utilized on the SLM were generated using code from MATLAB. This code was modified from code utilized in experimental work using LG beams (Ref. 4) and Gaussian Schell-model (GSM) beams (Ref. 5). Though the code to generate the GSM beams was used as the backbone for the generation of the FFIs, actual GSM beams were not used in this research.

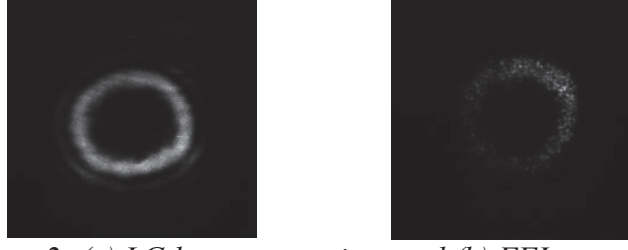


Figure 2: (a) LG beam at receiver and (b) FFI at receiver

Though similar at the receiver, as shown in Fig. 2, the SLM screens for the image generation are highly different for the LG beams and FFI. LG beams are coherent beams that carry OAM. This is in contrast to the FFIs, which are random and incoherent and lack phase information. The contrast between the two types of SLM screens can be seen in Fig. 3. These screens were each respectively used to generate the images shown in Fig. 2.

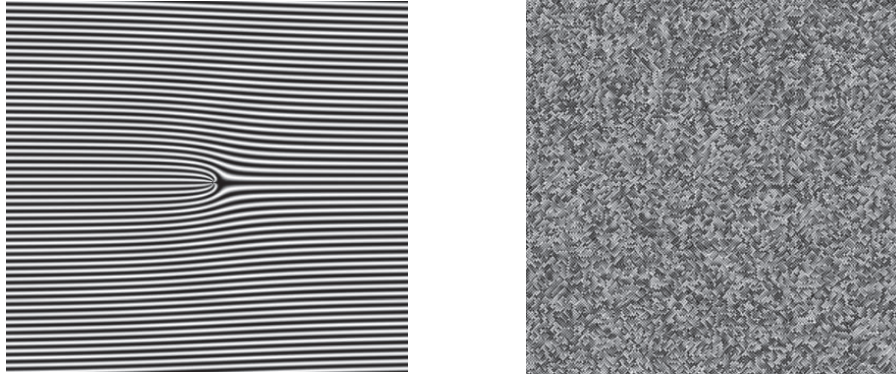


Figure 3: (a) LG beam SLM screen and (b) FFI SLM screen

3. EXPERIMENTAL SET UP

For both the LG beams carrying OAM and the FFIs, the laser beam was propagated through a 2.5-meter tank of water and projected onto a screen, where it was then captured using a high-speed camera, which can be seen in Fig. 4. The camera captured images at 320×320 pixels, and with 8 bits and 2x binning. This was done to keep the size of the images down for faster processing. The camera operated at 50 FPS and with a 19-millisecond shutter. 2500 images of each beam type were collected, accounting for 50 seconds of propagation through turbulence per collection.

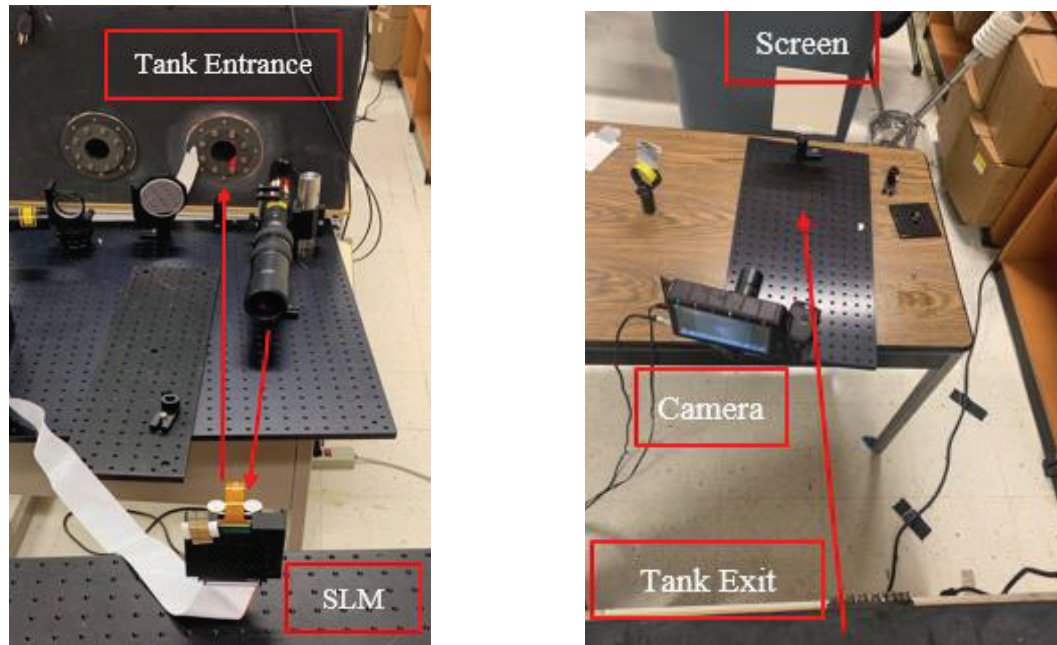


Figure 4: Experimental set-up

Originally, both types of structured light were directly propagated through turbulence, but in preliminary testing, it was found that the relative scale size of the turbulent eddies creating the optical turbulence and the beam width of the transmitted image had significant impact on the distortions seen in the light at the receiver. It was experimentally observed that the propagating light of the FFIs diverged at a much higher rate than the LG beams carrying OAM. This resulted in a large discrepancy in beam width at the point of turbulence, where the LG beam had a diameter of 0.7 centimeters and the FFI had a diameter of 1.5 centimeters, shown in Fig. 5.



Figure 5: (a) LG beam (0.7 cm diameter) and (b) FFI (1.5 cm diameter)

This was further validated by using a shadowgraph to characterize the turbulent flow of the heater used to create the thermal fluctuations that result in the refractive-index fluctuations. As can be seen in Fig. 6, the turbulent flow is on the scale size of approximately 1 centimeter. In this, an important assumption was made being that the temperature fluctuation profile followed the velocity fluctuation profile, which is what can be qualitatively observed in the shadowgraph. A

more accurate measurement would account for diffusion, but for the purpose of scale-size estimation, the velocity profile is deemed sufficient.

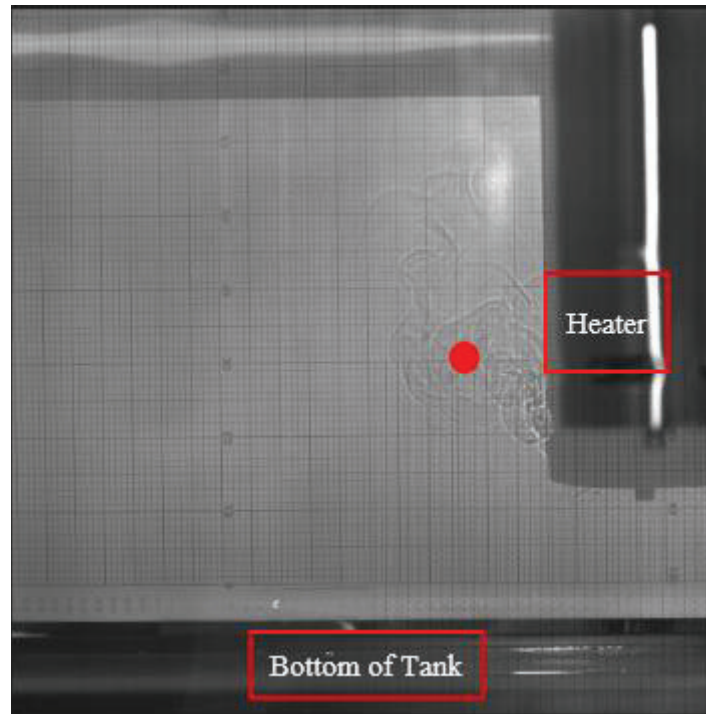


Figure 6: Heater turbulence characterization with approximate beam/image location shown

In order to correct this deficiency, a 4F lens system was utilized to partially-collimate and reduce in size the FFI. The lens system was comprised of a 400-millimeter lens and 300-millimeter lens separated by approximately 1 meter, shown in Fig. 7.

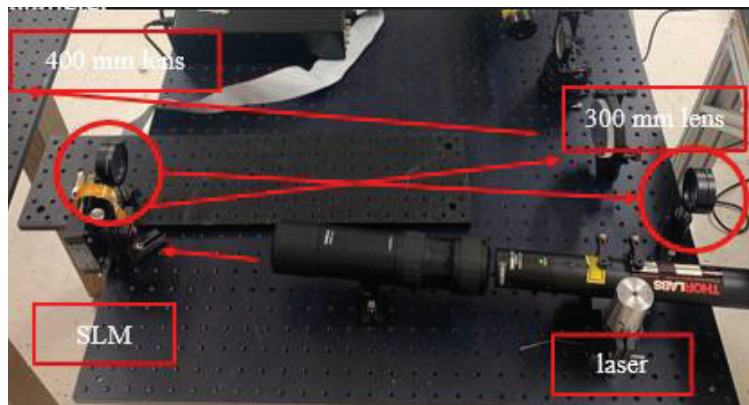


Figure 7: Added optical system for FFI to correct size discrepancy at turbulent point

Using the 4F lens system, the FFI at the turbulent point was decreased to 1-centimeter in diameter creating a much more similar turbulent effect on both the FFIs and LG beams, which were 0.7 centimeters in diameter (Fig. 5). The new FFI is shown in Fig. 8.

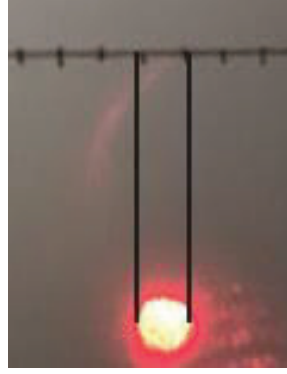


Figure 8: New FFI at turbulent point (1 cm diameter)

4. USE OF DEEP CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNNs) provide an effective and reliable method for classifying images of beams carrying OAM, even in high attenuation and optical turbulence. CNNs are a type of neural network specifically designed for image classification, as the traditional machine learning model of numbers as independent inputs and outputs cannot be used due to the spatial structure of the pixels containing necessary information for understanding the image as a whole. In a CNN, each layer takes in a set of images as the input, and produces a set of images as an output, preserving the proximity relationship of the pixels. CNNs are typically composed of many sets of three layers: a 2-D convolutional layer, a rectified linear unit layer, and a max pooling layer. The outputs from these layers are then used to classify the image in the output layers.

Layer	Type	Details
1	Input	64 x 64 grayscale image
2	2D Convolution	8 3 x 3 filters, stride 1, padding "same"
3	Batch Normalization	
4	Rectified Linear Units	
5	Max Pooling	2 x 2 max pooling, stride 2
6	2D Convolution	8 channels, 16 3 x 3 filters, stride 1, padding "same"
7	Batch Normalization	
8	Rectified Linear Units	
9	Max Pooling	2 x 2 max pooling, stride 2
10	2D Convolution	16 channels, 32 3 x 3 filters, stride 1, padding "same"
11	Batch Normalization	
12	Rectified Linear Units	
13	Fully Connected	3 outputs
14	Soft Max	
15	Classification	3 classes

Figure 9: CNN Architecture [3]

This architecture is based on the architecture used in Ref. 3 and consists of 15 layers in total. This particular architecture consists of an input layer, three sets of 2D-convolutional, rectified linear unit, and max pooling layers, and finishes with a 3-node fully connected layer. This is summarized in Fig. 9 above. The deep CNN designed, trained, and used to test the images using MATLAB 2021a and the Deep Learning Toolbox.

The images were preprocessed using a bicubic interpolation to resize the images from 320 x 320 pixels to 64 x 64 pixels. The images were reduced in size to speed up the training and classification speed of the CNN, an important consideration for a communication system. The CNN is only comprised of 15 layers for the same reason, as larger CNNs are generally more computationally complex. The images were randomly assigned to the training, validation, and testing sets, with 40% being used for training, 50% being used for testing, and 10% used for validation. Training used the stochastic gradient descent method with momentum of 0.9, with an initial learn rate of 0.01 and mini batch size of 64. The training was terminated after a maximum of 5 epochs.

5. RESULTS

After propagation through the optical turbulence, there was a large qualitative difference between how the turbulence affected the LG beams and FFIs. It is well-documented that the LG beams distort through stretching, intensity scintillation, and movement of the beam vortex [2-3]. This was reaffirmed through our own study. Interestingly, this was markedly in contrast with the way that the FFIs were affected by the changes to the refractive-index. Rather than shimmering and beam movement, the FFIs appears to disintegrate and reform as the optical turbulence passed through. The static difference between these affects can be seen in Fig. 10.



Figure 10: (a) FFI partially disintegrated by OT and (b) LG beam distorted by OT

2500 images of three patterns were collected for both the LG beams and FFIs, resulting in 15,000 total images. Two CNNs were trained, with one being solely on the images of the LG beams and the other on the FFIs. 1000 images of each pattern were used to train each CNN (3000 images total), and the remaining 4500 images were used in validation and testing. It was found that there was a 99.56% accuracy for the LG beam CNN and a 98.84% accuracy for the FFI CNN. The breakdown of which patterns were mistaken for other patterns can be seen in Fig. 11 and Fig. 12. From these results two important conclusions can be drawn. First, the CNN appears to prefer the type of distortion seen in the LG beams over the type of distortion of the FFIs. This is potentially due to the fact the CNN is able to learn the stretching, scintillation, and wander of the LG beams, but cannot learn the disintegration of the FFIs as well. This potential explanation is strengthened

by the confusion chart for both. For the LG beams (Fig. 11), the misclassifications consistently happened between two patterns (pattern 1 and pattern 3). This is in contrast to the FFIs, where misclassifications occurred between patterns 1 and 2, and then also patterns 2 and 3.

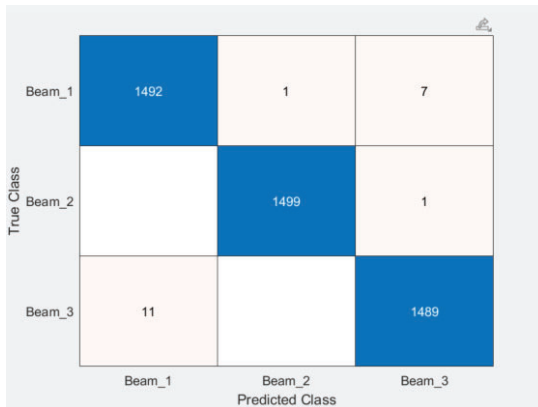


Figure 11: LG beam confusion chart

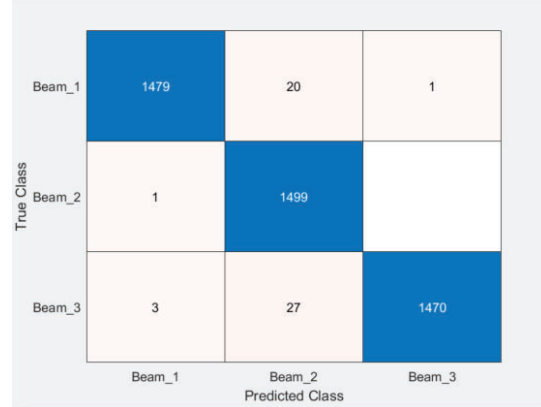


Figure 12: FFI confusion chart

However, it is important to note that not enough data nor enough patterns were compared to be able to definitively say for sure which the CNN “prefers”. Both the LG beam CNN and FFI CNN classified the images with very high accuracies, which provides further evidence for use of machine learning-based classification systems to be utilized in communication systems to decrease error rates.

6. CONCLUSION

The objective of this research was to investigate and compare the effects of thermally-generated optical turbulence on two forms of structured light, Laguerre-Gaussian beams and Far Field Images, by utilizing a deep convolutional neural network. The LG beams and FFIs were chosen due to their highly diverse properties of light, namely differences in the coherence, collimation, and orbital angular momentum. The beams were propagated underwater through a 2.5-meter tank, and were similar in size at the point of turbulence. It was found that the CNN preferred the LG beams over the FFIs, but further research is required to validate these findings. Additionally, the research provided more evidence for the effectiveness of neural network-based communication systems for decreasing error rates. Further research should look at increasing the number of patterns the CNN is forced to decide between (increasing complexity) and examining other forms of structured lights with more varying properties to widen the scope of the research.

7. REFERENCES

- [1] Doster, T., & Watnik, A. T., “Machine learning approach to OAM beam demultiplexing via convolutional neural networks,” *Applied optics*, 56(12), 3386-3396, 2017.
- [2] S. Avramov-Zamurovic, C. Nelson, and J. M. Esposito, “Using machine learning to classify laser light carrying orbital angular momentum in the presence of optical turbulence: an experimental study,” Submitted to *Optics Express*, (2020).

- [3] S. Avramov-Zamurovic et al, "Machine-learning-aided classification of beams carrying orbital angular momentum propagated in highly turbid water," J. Optical Society of America. (2020).
- [4] S. Avramov-Zamurovic, A. T. Watnik, J. R. Lindle, and K. Peter Judd, "Designing laser beams carrying OAM for a high-performance underwater communication system," Journal of the Optical Society of America. (2020).
- [5] Basu, Santasri, et al. "Computational approaches for generating electromagnetic Gaussian Schell-model sources." *Optics express* 22.26 (2014).

Appendix C: Conference on Lasers and Electro-Optics 2-Page Summary

Machine Learning-Based Underwater Communication System Using Light Carrying Orbital Angular Momentum

William A. Jarrett¹, Svetlana Avramov-Zamurovic¹, Joel M. Esposito¹, and Charles Nelson²

¹Weapons, Robotics, and Control Engineering Department, United States Naval Academy

²Electrical and Computer Engineering Department, United States Naval Academy

597 McNair Rd, Annapolis, Maryland 21401, USA

Author e-mail address: w.a.jarrett22@gmail.com

Abstract: An underwater communication system using light carrying orbital angular momentum is evaluated using a convolutional neural network through simulated and experimental thermally-generated underwater optical turbulence. 100% classification is demonstrated in no or weak turbulence. © 2022 W. A. Jarrett, S. Avramov-Zamurovic, J. M. Esposito, C. Nelson

1. Introduction

Light carrying orbital angular momentum (OAM) is of interest in optical communication systems for increasing system bandwidth. Specifically, we focus on a communication system utilizing Laguerre-Gaussian (LG) beams, where information is spatially multiplexed into the resulting intensity patterns of the superimposed beams, possible due to the orthogonality of light carrying OAM [1]. The information is decoded using a deep convolutional neural network (CNN). It was shown that CNNs are capable of highly accurate image classification, even in the presence of significant optical turbulence [2] and attenuation due to scattering and absorption [17], the primary obstacles in underwater wireless optical communication systems.

1.1. Spatial Multiplexing of Laguerre-Gaussian Beams

The proposed system uses N superimposed basis LG beams, each encoding a bit of information, to create an alphabet with 2^N unique intensity patterns. While previous work has evaluated 4-5 basis beams through optical turbulence [2], this investigation uses 8 basis beams, for a total alphabet size of 256, doubling the amount of information passed per message. With each additional basis beam, the alphabet increases exponentially, meaning a more complex problem for the CNN, even in the same environmental conditions. Methodology for the generation of Laguerre-Gaussian beams is given in [13]. Fig. 1 demonstrates how basis beams can be switched on and off to encode bits of information and generate an alphabet for practical implementation of this system. While Beam 35 is shown, any number of the basis beams can be superimposed at once, with each combination producing a unique image.

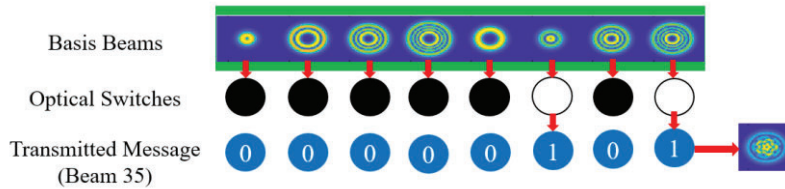


Fig. 1. Demonstration of 8 basis beam superposition for alphabet generation.

1.2. Deep Convolutional Neural Network

We use a 15-layer CNN comprised of an input layer, 3 sets of 2D-convolutional, rectified linear unit, and max pooling layers, a softmax layer, a fully-connected layer, and a classification layer. This design was selected to balance training / testing speed with classification accuracy, and is based on the architecture provided in [17]. The CNN was designed, trained, and tested in MATLAB 2021b using the Deep Learning Toolbox. The images were preprocessed using a bicubic interpolation to resize the images to 64 x 64 grayscale. Training used the stochastic gradient descent method with momentum of 0.9, with an initial learn rate of 0.01 and mini-batch size of 64. The training was terminated after a maximum of 10 failed validation checks.

2. Experimental Setup

Our setup is given in Fig. 2a with power supply (A), stabilized 2 mW, 632.8 nm He-Ne laser (B), beam expander (C), spatial light modulator (SLM) (D), mobile workstations (E_{1,2}), water tank (F), thermal generator (G), lens (H), display screen (I), and camera (J). The 256-beam alphabet (Fig. 2b) is generated using a 1920 x 1152 pixel SLM.

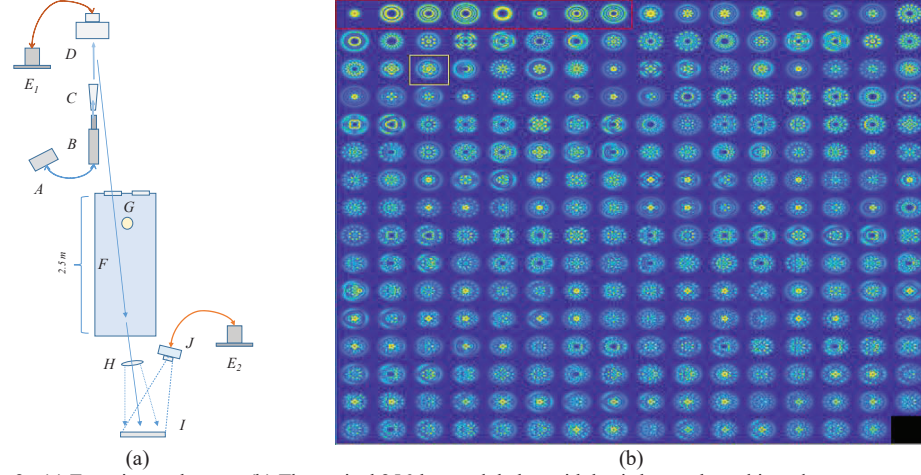


Fig. 2. (a) Experimental set-up (b) Theoretical 256-beam alphabet with basis beams boxed in red

3. Results

The 256-beam alphabet was propagated through simulated underwater optical turbulence, using thin phase screens randomly drawn from the Nikishov power spectrum with a rate of dissipation of kinetic energy $\varepsilon = 1 \times 10^{-5} [\frac{m^2}{s^3}]$ and a rate of dissipation of mean-squared temperature $\chi_T = 1 \times 10^{-7} [\frac{K^2}{s}]$ [29]. The primary parameters for the simulation were a screen resolution of $N = 1024 \times 1024$ pixels, size of the transmitter and receiver $D = 0.1$ m, total propagation distance $\Delta z = 30$ m, wavelength of the light $\lambda = 635$ nm, the beam waist radius $\omega_0 = 0.01$ m, and the number of phase screens along the path of propagation $n = 10$. The CNN achieved a classification accuracy of 100% with 100 images per beam used for training (20% of total set) and 400 images per beam used for testing. The alphabet (Fig. 2b) was then propagated through quiescent water (Fig. 2a), achieving a 100% classification accuracy with 200 images per beam used for training (20% of total set) and 800 images per beam, or 204,000 images, used for testing. Fig. 3 is given to show the similarity between the theoretically produced images and experimental images.



Fig 3. (a) Beam 35 in simulated underwater weak optical turbulence (b) Beam 35 experimentally captured in quiescent water (boxed in yellow in Fig. 2b)

We demonstrate the CNN has no difficulty classifying the 256-beam alphabet in no or weak optical turbulence, providing a proof-of-concept for the proposed CNN-based underwater wireless communication system, for high-bandwidth reliable operation. We are preparing the experiment to propagate the 256-beam alphabet through strong optical turbulence as outlined in [2], and will report our results at the conference. From the work done in [2], it is predicted that a classification accuracy of 99.9% can be achieved, matching the bit error rating (~ 7 errors per 10,000 images) of the 32-beam alphabet.

Appendix D: 2022 Optica Imaging and Applied Optics Conference 2-Page Summary

Challenges when Partially Training a Machine Learning-Based Optical Communication System in Variable Experimental Conditions

William A. Jarrett¹, Svetlana Avramov-Zamurovic¹, Joel M. Esposito¹, and Charles Nelson²

¹Weapons, Robotics, and Control Engineering Department, United States Naval Academy

²Electrical Engineering Department, United States Naval Academy

597 McNair Rd, Annapolis, Maryland 21401, USA

Author e-mail address: m223222@usna.edu

Abstract: We present challenges when training a machine learning-based underwater wireless optical communication system in selected experimental scenarios. The system is tested under different conditions, that include minor beam misalignment and varying optical turbulence. © 2022 W. A. Jarrett, S. Avramov-Zamurovic, C. Nelson, J. M. Esposito

4. Introduction

Light carrying orbital angular momentum (OAM) is of interest in optical communication systems for increasing system bandwidth. Specifically, we focus on a communication system utilizing Laguerre-Gaussian (LG) beams, where information is spatially multiplexed into the resulting intensity patterns of the superimposed beams, possible due to the orthogonality of light carrying OAM [1]. The information is decoded using a deep convolutional neural network (CNN). It was shown that CNNs are capable of highly accurate image classification, even in the presence of significant optical turbulence [2] and attenuation due to scattering and absorption [3], the primary obstacles in underwater wireless optical communication systems. However, CNNs are challenged significantly when used to classify the same patterns, but under differing experimental conditions. This work seeks to understand these challenges and provide a methodology for designing and training a robust CNN.

4.1. Spatial Multiplexing of Laguerre-Gaussian Beams

The proposed system uses N superimposed basis LG beams, each encoding a bit of information, to create an alphabet with 2^N unique intensity patterns. With each additional basis beam, the alphabet increases exponentially, meaning a more complex problem for the CNN, even in the same environmental conditions. Methodology for the generation of Laguerre-Gaussian beams is given in [4]. Fig. 1 demonstrates how basis beams can be switched on and off to encode bits of information and generate an alphabet for practical implementation of this system. While Beam 5 is shown, any number of the basis beams can be superimposed at once, with each combination producing a unique image.

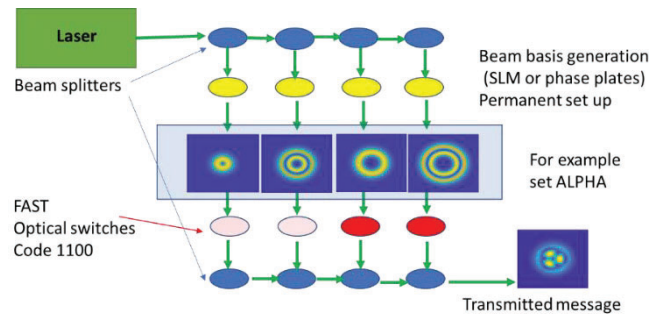


Fig. 1. Demonstration of 4 basis beam (alphabet ALPHA) superposition for alphabet generation.

4.2. Deep Convolutional Neural Network

We use a 15-layer CNN comprised of an input layer, 3 sets of 2D-convolutional, rectified linear unit, and max pooling layers, a softmax layer, a fully-connected layer, and a classification layer. This design was selected to balance training / testing speed with classification accuracy, and is based on the architecture provided in [3]. The CNN was designed, trained, and tested in MATLAB 2021b using the Deep Learning Toolbox. The images were preprocessed using a bicubic interpolation to resize the images to either 277x 277 or 64 x 64 grayscale. Training used the stochastic gradient descent method with momentum of 0.9, with an initial learn rate of 0.01 and mini-batch size of 64. The training was terminated after a maximum of 10 failed validation checks.

5. Experimental Setup and Data Collection

The experimental setup follows the same setup outlined in [2], utilizing a 1920 x 1152-pixel SLM to generate the LG beams, a heater to generate optical turbulence in a tank of water, and a camera to capture the images for processing and CNN classification. This setup allows us to maintain similar turbulent conditions between trials [2]. For this experiment, alphabet ALPHA was propagated through variable conditions, summarized in Table 1. For each collection, 1000 images of each beam type were taken, resulting in a total of 90,000 images. These images were then processed in MATLAB 2021b to minimize background noise and increase contrast, resulting in the selected images shown in Fig. 2.

Table 1. Variable Experimental Conditions

Dataset	Translation	Turbulence
A	Center	Off
B	Center	On
C	Right	Off
D	Right	On
E	Down	Off
F	Down	On

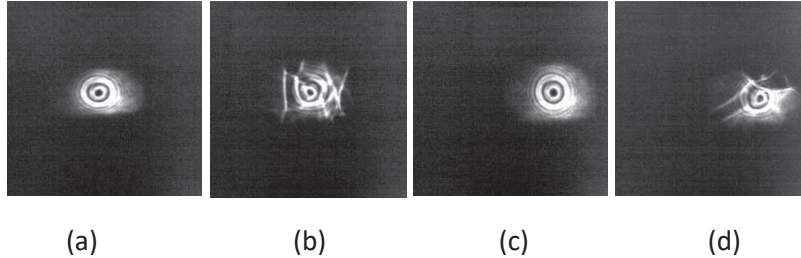


Fig. 2. (a) Beam 2 in quiescent conditions, center. (b) Beam 2 in quiescent conditions, translated right. (c) Beam 2 in turbulent conditions, center. (d) Beam 2 in turbulent conditions translated right.

6. Results

Preliminary testing trained our CNN on Dataset B using 277 x 277-pixel images, summarized in Table 2.

Table 2. Preliminary Testing Results

Training Set	Testing: A	B	C	D	E	F
B	16.36%	99.08%	21.56%	9.70%	6.67%	5.73%

As evident, the CNN only performed to a satisfactory level when tested on material used in training. We will report on the training and testing accuracies of the CNN for each of the different datasets. We will then investigate and develop a methodology for CNN training for the better practical use in image-based underwater wireless optical communication systems, ensuring systems are resilient to a robust variety of conditions.

Appendix E: 2022 SPIE Optics and Photonics Abstract

**Experimental Propagation of Light Carrying Orbital Angular Momentum
Through Turbulent Rayleigh-Bénard Convection Utilizing Machine
Learning-Based Classification**

W. A. Jarrett, S. Avramov-Zamurovic, J. M. Esposito, K. P. Judd, S. Matt, R. A. Handler, A. T. Watnik, J. R. Lindle, and C. Nelson

Abstract: (250 words)

Light carrying orbital angular momentum (OAM) has garnered recent interest in communication systems as it can be multiplexed for achieving a higher bandwidth. This investigation focuses on a communication system that utilizes Laguerre-Gaussian (LG) beams carrying OAM, with information spatially multiplexed into the resulting intensity patterns of the light [1]. The information is then decoded using a deep convolutional neural network (CNN). It was shown that CNNs are capable of high levels of image classification accuracy, even in the presence of significant optical turbulence [2] and attenuation due to scattering and absorption [17]. The proposed system uses N LG basis beams, combined orthogonally, to create an alphabet of unique intensity patterns the size of 2^N . While previous work has evaluated 4-5 basis beams through optical turbulence [2], this investigation uses 9 basis beams, for a total alphabet size of 512, more than doubling the amount of information passed per message. As the size of the alphabet increases exponentially, each additional basis beam creates a more complex problem for the CNN. The 512-beam alphabet will be experimentally evaluated by passing the beams through turbulent Rayleigh-Bénard convection generated in a water channel, as this is a type of well-documented turbulent flow that can be created in a controlled experimental set-up [48]. This allows for investigation of the performance of a novel communication system through realistic underwater turbulence, while reporting on the effects of optical turbulence on beams carrying OAM. System performance will be quantified using the classification accuracy of the CNN.