Turbulence Mitigation with Deep Learning for Image Recognition

David Vint

A thesis submitted for the degree of

Doctor of Philosophy

Neuromorphic Sensor Signal Processing Lab

Centre for Signal and Image Processing

Department of Electronic and Electrical Engineering

University of Strathclyde

Glasgow

2026

Declaration

Signed: David Vint
Date: 05/02/2026

Acknowledgement

I would firstly like to extend my deepest thanks to my PhD supervisor, Dr Gaetano Di Caterina. He has stuck by me throughout the highs and lows of my studies, and was always there when I needed help or assistance. As his first PhD student, it cannot have been easy, and I thank him very much for his patience.

I would also like to thank my industrial supervisor, Robert Lamb, who always provided me with useful feedback, as well as his extensive knowledge in the field of optics. This thanks also extends to David Humphreys, who I hope is enjoying retired life.

Next I would like to thank my lab partners: Dr Kirkland, Dr Manna and Dr Vicente Sola. I cannot imagine getting where I am today without you three, you have not only been incredible 'coleegs' but have also become dear friends, I cannot thank you enough.

Next to thank is my family. My mother and father: Sally and Toby and my brothers: Neil and Peter (And the dogs). Another group of people that I would not be here without. Throughout my time as a student, there have been highs and some extreme lows (Looking at you Covid-19). The love and support I have received from my family throughout is something I cannot overstate, and I hope that I did not represent a pantechnicon of stress.

I would also like to extend my deepest gratitude to Geraldine, who helped me come back from some tough spots, and has helped me find the positives in life.

Finally, I would also like to thank all of my friends in Glasgow (and Stafford). There will soon be a time that you will no longer need to ask me 'When are you going finish your PhD?'. Special thanks goes to Matt and Heather, who have always supported me and distracted my studies with games of Dominion.

Abstract

The classification and tracking of targets is a task commonly performed over large distances, and it can be impacted by atmospheric turbulence. As light travels from the target, towards the camera lens, it is distorted due to fluctuations in the refractive index of air. When incident on a camera lens, such a distorted wave results in blur and warping of the final image, therefore increasing the difficulty of post-acquisition tasks such as tracking. It is therefore desirable for such distortions to be minimised.

Whilst traditional turbulence mitigation techniques are able to improve the quality of such images, the ever increasing availability of deep learning has allowed huge progress in many computer vision and image processing applications. This therefore prompts an investigation into its applicability for turbulence mitigation. This thesis presents such an investigation.

In order to train deep learning networks, a substantial dataset is required. However, due to difficulties in data acquisition for turbulence mitigation, simulation tools are an obvious alternative to apply turbulence effects onto clean images. This not only allows any image contents to be used, but also allows full control over the atmospheric conditions.

This thesis presents the development of a turbulence simulator, where the light from a point source is propagated through a simulated atmosphere using a split-step propagation method. Using this simulator, several datasets were synthesised and used throughout the remainder of the work.

Leveraging the data generated through the simulator, an investigation into the applicability of deep learning for turbulence mitigation was undertaken by utilising existing off-the-shelf deep learning architectures not originally designed for turbulence mitigation. By retraining these networks on turbulent data, their ability to shift domain was tested, where it was found that deep learning could indeed be applied to turbulence mitigation; however more success would be gained by using video sequences as opposed to single images. With this information, two video processing models were tested, EDVR and DATUM. It was found that, by altering the loss function of these models to prioritise perceptual image quality, post-mitigation classification could be greatly improved.

In this context, it was also found that turbulence mitigation literature over relies upon common image quality metrics such as PSNR and SSIM. As tasks such as classification and tracking are the motivation behind turbulence mitigation, in this research it is argued that such metrics are ill-suited for turbulence mitigation algorithm quality assessment. Therefore, to gain an understanding of which metrics would be better suited for this task, an investigation was undertaken, where metric scores were compared with classification accuracy. The metrics that best aligned with this accuracy were identified as the optimal metrics for turbulence mitigation quality assessment. In this, it was found that the DISTS metric was the best full-reference metric, whilst Q-ALIGN was found to be the best no-reference metric.

The full scope of this thesis therefore covers the varied aspects of turbulence mitigation research. From data sparsity and simulation, to removal and quality assessment.

Contents

Acronyms

ADAM	Adaptive Moment Estimation
AGGD	Asymmetric Generalised Gaussian Distribution
AHIQ	Attention-based Hybrid Image Quality metric
AI	Artificial Intelligence
AIM	Advances in Image Manipulation
ANN	Artificial Neural Network
ARNIQA	leArning distoRtion maNifold for Image Quality Assessment
ATSyn	Atmospheric Turbulence Synthesis dataset
BAcc	Balanced Accuracy
BRISQUE	Blind/Referenceless Image Spatial Quality Evaluator
CAE	Convolutional Autoencoder
CLEAR	Complex waveLEt fusion for Atmospheric tuRbulence
CLIP	Contrastive Language–Image Pretraining
CLIPIQA	CLIP Image Quality Assessment
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
CSA	Cross-Scale Attention
CSIQ	Categorical Subjective Image Quality
CVPR	Computer Vison and Pattern Recognition
CW-SSIM	Complex Wavelet Structural Similarity
DAAB	Deformable Attention Alignment Block
DAN	Deep Attention Network
DATUM	Deep Atmospheric Turbulence Mitigation
DCT	Discrete Cosine Transform
DConv	Depthwise Convolution
DDPM	Denoising Diffusion Probabalistic Models
DIM	Deep Integration Module
DISTS	Deep Image Structure and Texture Similarity
DL	Deep Learning

DnCNN	Denoising Convolutional Neural Network
DNN	Deep Neural Network
DOF	Depth of Feild
DT-CWT	Dual Tree Complex Wavelet Transform
DWT	Discrete Wavelet Transform
EDVR	Enhanced Deformable Video Restoration
FATMOSE	False Bay ATMOSpheric Experiment
FFHQ	Flickr-Faces-HQ
FN	False Negative
FP	False Positive
FR	Full Reference (image quality metrics)
FSIM	Feature SIMilarity Index
FSIMc	Feature SIMilarity chrominance
GAN	Generative Adversarial Network
GDFN	GatedDconv feed-forward network
GGD	Generalised Gaussian Distribution
GLP	Gated Local Pooling
GPU	Graphics Processing Unit
GRUM	Gated Reference Update Module
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IR	Image Recovery
IRIS	Image Restoring and Image Subsampling
IQA	Image Quality Assessment
KRCC	Kendall's Rank Correlation Coefficient
LE	Long Exposure
LED	Light Emitting Diode
LIQE	Language-Image Image Quality Evaluator
LLM	Large Language Model
LPIPS	Learned Perceptual Image Patch Similarity
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAMAT	MAMba-based Atmospheric Turbulence removal
MANIQA	Multi-dimension Attention Network for Image Quality Assessment
MDTA	multi-Dconv head Transposed attention
ML	Machine Learning
MLP	Multilayer Perceptron
MOS	Mean Opinion Score

MSCN	Mean Subtracted Contrast Normalised
MS-SSIM	Multi-Scale Structural Similarity
MSE	Mean Squared Error
MTF	Modulation Transfer Function
MVG	Multivariate Gaussian
NIQE	Natural Image Quality Evaluator
NR	No Reference (image quality metrics)
NSS	Natural Scene Statistics
NTIRE	New Trends in Image Restortation and Enhancement
OTF	Optical Transfer Function
OTIS	Open Turbulent Image Set
PCA	Principle Component Analysis
PCD	Pyramid, Cascading and Deformable convolution
PD	PreDeblur
PI	Perceptual Index
PIPAL	Perceptual Image Processing ALgorithms
PIQUE	Perception-based Image QUality Evaluator
PieAPP	Perceptual image-error Assessment through Pairwise Preference
PLCC	Pearson Linear Correlation Coefficient
PPTRN	Probabilistic Prior Turbulence Removal Network
PSD	Power Spectral Density
PSF	Point Spread Function
PSNR	Peak Signal-to-Noise Ratio
RCAB	Residual Channel Attention Block
RCAN	Residual Channel Attention Network
RDB	Residual Dense Block
RDN	Residual Dense Network
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RIR	Residual In Residual
RNN	Recurrent Neural Network
RPCA	Robust Principal Component Analysis
SA	Self-Attention
SE	Short Exposure
SE PSF	Short Exposure Point Spread Function
SGD	Stochastic Gradient Descent
SGL	Sobolev Gradient and Laplacian

SOA	State Of the Art
SRCC	Spearman's Rank Correlation Coefficient
SSIM	Structural SIMilarity Index
SSPD	Sensor Signal Processing for Defence
SSTB	Scale Swin Transformer Block
SVM	Support Vector Machine
SW-MSA	Shifted Window-based Multi-head Self-Attention
TAB	Transposed Attention Block
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TRN	Turbulence Removal Network
TQI	Turbulence Quality Index
UHD	Ultra High Definition
VGG	Visual Geometry Group
ViT	Vision Transformer
WF	Wiener Filter

Chapter 1

Introduction

Optical imaging systems are constantly improving, especially in the case of long distance observations, where modern lenses can capture high quality images over extreme distances [1]. This capability is extremely beneficial for applications such as long distance object recognition [2], agriculture optimisation [3] and astronomy [4]. In such applications however, image distortions are still present due to diffraction limitations and atmospheric turbulence [5]. In both cases, the level of control is extremely limited, as the physics of diffraction cannot be overcome, and the amount of turbulence in the atmosphere cannot be controlled. In the case of diffraction, the distortion is spatially invariant and is a known quantity. The removal of its effects is therefore possible via deconvolution [6] and super resolution techniques [7].

The effects of atmospheric turbulence on the other hand are more severe, and less predictable. A turbulent atmosphere is caused by temperature, pressure changes and wind currents, which result in random fluctuations in the refractive index of air [8]. In the case of imaging through turbulence, light propagating through the atmosphere can become severely distorted, resulting in imagery that contains a significant amount of spatial blurring and warping. These distortions therefore increase the difficulty for tasks such as tracking or classification, and would ideally be mitigated beforehand. A possible approach to reduce this problem might be adjusting the imaging parameters, such that the turbulence distortion is minimised. For example, one could move closer to the target or wait for more favourable atmospheric conditions. However, in many applications, such solutions are not possible, thus requiring turbulence mitigation techniques.

Due to the spatially varying nature of turbulence distortion, the task of turbulence mitigation is extremely complex and difficult, and can be approached in two ways. The first is the use of adaptive optics, which aims to reduce turbulence at the time of image capture [5, 9, 10]. The second is the application of a post-acquisition processing step, which aims to recover a clean, turbulence-free, image. Of these two approaches, it is the post-acquisition turbulence mitigation that is of most interest, as it allows the recovery of distorted images, no matter what hardware was used for capture. Techniques for adaptive optics often require complex and expensive hardware [11], and are therefore less useful to a majority. The application of post-acquisition turbulence mitigation, on the other hand, can be achieved on simple hardware and is a well researched field [12–18].

In 2012, AlexNet, a deep learning model proposed by Krizhevsky et al. [19], won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), marking a breakthrough for convolutional neural networks (CNNs) in computer vision [20]. The increasing availability of GPU computing further accelerated interest in deep learning (DL), which now underpins state-of-the-art methods in tasks such as image classification, segmentation, deblurring, and super-resolution [7, 21–23].

Given the continued success of DL for machine vision, its use for turbulence mitigation was a natural progression. Where it was found that Machine Learning algorithms could indeed recover a clean image when provided with turbulence data, and outperform traditional approaches. Where the term 'traditional' is used throughout this thesis to denote earlier, image processing based, turbulence mitigation algorithms that are not data-driven (i.e. before AI/ML). Given this, the logical next step was the optimisation of this process, identifying the best model architectures and datasets that would lead to state-of-the art performance. This thesis presents such an investigation into the optimisation of of deep learning for turbulence mitigation, and comments on the best practices.

1.1 Research Motivation

Atmospheric turbulence imposes a fundamental limit to the angular resolution of many optical systems [5], especially in the case of long-range observations. The aberrations acquired during propagation through the atmosphere present as spatial warp and blur in captured images and videos. Such distortions therefore greatly hinder post-acquisition steps such as tracking or classification [24].

Traditional methods for turbulence mitigation in images and videos are mathematically complex, and have limited capability. With the ever growing success of deep learning models for image processing, its applicability to the task of turbulence mitigation was the next logical step, providing improved performance. Due to the early stages of such an application, a great deal of further improvement may be possible, greatly benefiting the turbulence imaging community.

In order to train deep learning networks, a large amount of diverse and accurate data is first required. While traditional methods are able to process images based on mathematical algorithms, and can therefore be applied to any turbulent image, deep learning relies on examples of both clean and turbulent data for the network to understand the underlying distortions and how to mitigate them. However, due to lack of control of the atmosphere, the acquisition of ground truth images, required for deep learning, is not possible.

The common solution when faced with a paucity of data is that of simulation. However, due to the complex nature of turbulence, such simulations are difficult to develop. Initial simulation tools approximate turbulence distortion with random warp and blurring, which does not represent an accurate model of turbulence [25]. More accurate simulation tools proposed in the literature have been developed exclusively for internal use, are not open source, or were created after the start of this research. There was therefore a significant lack in suitable data for deep learning, as the ability to synthesise turbulence data was not possible without accessible simulation techniques.

When training deep learning networks, a loss function is used to inform the model of its quality. Using this, the internal weights can be updated accordingly in order to improve performance. There is a great choice of loss functions in literature [26], however the choices made for image processing tasks such as deblurring or super resolution prioritise reconstruction error, as opposed to 'perceptual image quality'. In the case of turbulence mitigation, the most likely motivation for processing is that of tracking or classification. Therefore, loss functions that aim for reconstruction quality may not be suitable for this task.

Furthermore, image processing tasks such as deblurring make use of image quality metrics to assess model performance [27]. The commonly used metrics however are not suited for the field of turbulence mitigation as, similarly to the loss function, the purpose of turbulence mitigation is that of facilitating post-processing steps. The metrics with which images are measured should therefore be in line with this goal and reflect how well they can be processed, as opposed to how accurately they can be reconstructed.

In line with the potential misuse of metrics for turbulence research, it is also true that, across literature, common results are difficult to obtain. As different works make use of different metrics and different datasets, it is therefore extremely difficult to identify the best methods, as cross comparison is not possible.

1.2 Summary of Original Contributions

The main research contributions of this thesis are described below.

- **Development of a split-step propagation turbulence simulation tool.** Given the lack of publicly available simulation tools, this thesis presents the development of a turbulence simulator, created in MATLAB, that is able to apply the effects of turbulence onto clean images. The simulation tool is applicable to both single images and video sequences and is controlled by an extensive set of parameters, where many different aspects of the atmosphere can be altered for specific use cases. The development of this simulator solves the initial concerns, present at the start of this research, regarding the lack of available turbulence simulation techniques. (Chapter 4)

- **Synthesis of a new dataset of turbulence videos.** Using the developed simulator, a dataset was synthesised from the Places dataset [28]. The resulting dataset, 'turbulence videos', contains a total of 148,884 videos, with 15 frames per video, and has been made publicly available [29]. This dataset further provides a solution to the lack of available turbulence data, and can be used to train deep learning architectures in future literature. (Chapter 4)

- **Identification of the optimal deep learning architecture for single image turbulence mitigation.** Using the data from the simulator, an initial study was undertaken into the use of deep learning for turbulence mitigation. In this, several deep learning models from different domains, such as deblurring or super resolution, were re-trained on turbulence data. The goal of this study was to identify which model architecture would best adapt to turbulent data, providing the best results. Once tested, it was found that the majority of the deep learning networks were indeed able to perform turbulence mitigation, and the best performing architectures are highlighted.(Chapter 5)

- **Optimisation of the loss function for turbulence mitigation models.** By training two modern models with a perceptual loss element, it was found that the post-processing capability of images/videos could be greatly improved. This demonstrates that, when training deep learning networks for turbulence mitigation, the post-mitigation application should always be taken into consideration. This finding demonstrates how future deep learning algorithms should be trained, and therefore is a vital contribution to future research. (Chapter 5)

- **Analysis of image quality metrics for turbulence mitigation.** It is shown that the use of commonly used image quality metrics is not suitable for turbulence mitigation quality assessment, as such metrics aim to optimise reconstruction quality, as opposed to perceptual quality. A study was undertaken to identify the optimal full-reference and no-reference metrics for turbulence data. It is also shown that when measuring the quality of such models, it is best to test on application specific data, i.e. if the goal is classification, then classification should be the quality metric. This study presents a deeper understanding into how turbulence mitigation algorithms should be compared, and makes suitable recommendations for future works. (Chapter 6)

- **Development of new metrics for turbulence mitigation quality assessment.** Using the results from the investigation into image quality metrics, two new metrics are proposed, TQI and TQI_T. These metrics combine other metric scores to provide a wider understanding of image quality after turbulence mitigation.(Chapter 6)

1.3 Organisation of the thesis

The remainder of the chapter is organised as follows.

Chapter 2 firstly covers the required knowledge for turbulence simulation, including the theoretical background for modelling turbulence and the method used for propagation through the atmosphere. The existing literature on turbulence data, simulation and mitigation is then presented.

Chapter 3 provides information regarding deep learning, including commonly used techniques and their applications. Existing deep learning techniques for turbulence mitigation are also reviewed, alongside an extensive analysis of image quality metrics.

Chapter 4 then provides the details of the turbulence simulator and its development. Firstly it explains how each separate element functions, followed by the description of the simulator as a whole. The techniques used to validate the simulator are then provided, showing that the resulting distortions match with the expected theory. The several datasets that were synthesised with the simulator are then detailed.

Deep learning is then applied to the task of turbulence mitigation in Chapter 5, where an initial study is performed using off-the-shelf architectures from other domains. Using the information from this study, the investigation was expanded to the use of video sequences. It is found that by adapting the loss function for post-mitigation classification, a greater performance can be obtained. This finding is then adapted to a current state-of-the-art model, where it is again found that the loss function can be optimised towards turbulence mitigation applications.

Chapter 6 details an investigation into the use of image quality metrics for turbulence mitigation, for which it is found that commonly used metrics are ill-suited. With this finding, multiple metrics were then tested on common distortions in order to identify the behaviour of each. A similar process is then applied to turbulence mitigated images, to identify which would correlate positively with classification results.

The conclusion of the thesis is then presented in Chapter 7 alongside possible future directions of research.

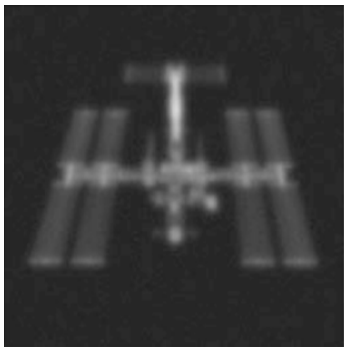
Chapter 2

Turbulence Theory and Applications

2.1 Introduction

Atmospheric turbulence is caused by random fluctuations in the refractive index of air [8]. These fluctuations can cause wavefronts to warp and distort as they travel, resulting in difficulties for numerous signal processing tasks such as image/video acquisition [30], communications [31] or classification/recognition tasks [24]. Such applications commonly require observations over a large distances, a characteristic that greatly increases the potential of atmospheric turbulence interference. Other factors that can increase the difficulty of these tasks include observation through different atmospheric layers (air-to-ground or ground-to-air) or 'close-to-ground' imagery, which is subject to more varied heat fluctuations [12]. Whilst the mitigation of turbulent distortion can be achieved using adaptive optics [5, 9, 10], such techniques require complex and expensive hardware [11]. This thesis therefore focuses on post-acquisition processing techniques designed to mitigate these distortions, which affect numerous domains, from astronomy [4] to surveillance [32].

Due to the complex nature of the earth's atmosphere, the observation of stars and satellites can be significantly degraded by poor atmospheric conditions [33, 35]. In order to improve such observations, multiple algorithms have been developed to mitigate the atmospheric distortions [36, 37], therefore facilitating tasks such as target identification or understanding the working state of targets [4]. Another common use for turbulence mitigation is that of facial recognition [11, 38–40], where distortions can render a face unrecognisable. Similarly, tracking and classification can be severely impacted by atmospheric turbulence [24],


(a) Satellite observation [33].

(b) Facial recognition[11].

(c) Object detection [34].

Figure 2.1: Example domains in which atmospheric distortion hinder performance.

especially when operating over large distances. In such cases, the easy solution of 'moving closer to the target' may not be feasible, therefore requiring a post processing solution. Examples of atmospheric turbulence reducing the image quality in these various domains can be seen in Figure 2.1.

In order to develop algorithms for turbulence mitigation, it is first vital to define a theoretical model of turbulence. However, constructing such a model has proven challenging due to the extremely complex nature of turbulence, with mathematical difficulties arising in the solving of the Navier-Stokes equations [41–43]. Despite these challenges, researchers have formulated statistical theories by observing turbulent fields and analysing how turbulence originates and dissipates [41, 44, 45]. Using such theories, turbulence can be modelled and represented, therefore facilitating the continued development of mitigation algorithms.

Due to the uncontrollable nature of atmospheric turbulence, the acquisition of suitable data is a difficult task. Firstly, the number of factors that influence the atmosphere is substantial, including observation distance, observation angle, temperature and wind levels. Furthermore, the domain of the data in question may be extremely specific (i.e. images captured in an urban area, or in the desert). Therefore, in order to represent such a diverse dataset, capturing all potential scenarios, the logistics of data capture would be infeasible. Secondly, many algorithms require a turbulent free version of each image/video captured, typically referred to as 'ground truth'. This is also infeasible, as to obtain such an image, one would need to be able to 'turn off' the turbulence to capture a turbulent free image. It is for these reasons that, using the theoretical model as a foundation, simulation models have been developed that are able to apply turbulent distortion onto any scene required. This allows parameters to be varied, and applied to any images required, therefore allowing extreme flexibility for the development of mitigation algorithms.

The remainder of the chapter is organised as follows. Section 2.2 first presents the necessary background theoretical knowledge for turbulence simulation and mitigation. Section 2.3 then presents a review of real datasets captured for various purposes including algorithm testing and atmosphere characterisation. The different approaches to turbulence simulation are then presented in Section 2.4, as well as the datasets synthesised and made publicly available. Section 2.5 then provides a background into traditional (i.e. not based on deep learning) methods of turbulence mitigation. Section 2.6 concludes the chapter.

2.2 Turbulence Theory

2.2.1 Image Formation

When capturing images or videos through a camera system, there are two key factors that can affect the resulting image: the camera setup and the atmospheric conditions. The quality of the overall system can be described by the Optical Transfer Function (OTF) of each individual component. The OTF is a complex valued function that describes how an optical system transfers spatial frequencies from one plane to another. In the case of the atmosphere, this OTF, $H_{atm}\left(f_x, f_y\right)$, governs the frequency response between the object and the camera lens. In the case of the camera setup, the OTF, $H_{lens}\left(f_x, f_y\right)$, details the frequency response of the lens itself. The OTF of a full system can then be found via simple multiplication of the separate elements as,

$$H\left(f_x, f_y\right) = H_{atm}\left(f_x, f_y\right) H_{lens}\left(f_x, f_y\right) \tag{2.1}$$

At this stage, it is beneficial to define two key operators. These are the Fourier transform and the inverse Fourier transform, defined as,

$$\begin{aligned} \mathcal{F}\{U(\boldsymbol{r})\} = U(\boldsymbol{f}) &= \int_{-\infty}^{\infty} U(\boldsymbol{r})\, e^{-2\pi i f r}\, dr \\ \mathcal{F}^{-1}\{U(\boldsymbol{f})\} = U(\boldsymbol{r}) &= \int_{-\infty}^{\infty} U(\boldsymbol{f})\, e^{2\pi i f r}\, df \end{aligned} \tag{2.2}$$

To transfer the OTF to the spatial domain, the inverse Fourier transform can be applied to generate a Point Spread Function (PSF), as

$$h(x,y) = \mathcal{F}^{-1}\left\{H\left(f_x, f_y\right)\right\}, \tag{2.3}$$

where the PSF describes how light from a point source spreads as it propagates through an optical system. For simple blurring, for example, the PSF would resemble a smooth, symmetric spot that spreads outward from the centre. In this case, the PSF may be spatially invariant. i.e. the PSF would be the same for each pixel within the image. However, more complex imaging systems may have spatially varying blur, which would result in a separate PSF for each pixel. The application of a PSF to a clean image, I, can be achieved via convolution in the spatial domain as,

$$\hat{I}(x,y) = (I * h)(x,y) = \iint I(f_x,f_y)\, h(x - f_x,\, y - f_y)\, df_x\, df_y, \tag{2.4}$$

where, $\hat{I}$ is the resulting distorted image. Equivalently, the same action can be achieved in the frequency domain via multiplication as,

$$\hat{I}(x,y) = \mathcal{F}^{-1}\big\{I(f_x,f_y)\, H(f_x,f_y)\big\} \tag{2.5}$$

For spatially variant PSFs, each would need to be applied to its corresponding pixel, therefore resulting in the following convolution,

$$\hat{I}(x,y) = (I * h)(x,y) = \iint I(f_x,f_y)\, h_{x,y}(x - f_x,\, y - f_y)\, df_x\, df_y \tag{2.6}$$

where no frequency domain counterpart exists for spatially varying PSFs.

Given the definition in Eq. (2.5), it can be seen that, if the system's OTF was known, the frequency content of the original image could be recovered as,

$$I(f_x,f_y) = \frac{\mathcal{F}\big\{\hat{I}(x,y)\big\}}{H(f_x,f_y)} = \frac{\hat{I}(f_x,f_y)}{H(f_x,f_y)} \tag{2.7}$$

allowing the clean image to be reconstructed as,

$$I(x,y) = \mathcal{F}^{-1}\left\{I(f_x,f_y)\right\} = \mathcal{F}^{-1}\left\{\frac{\hat{I}(f_x,f_y)}{H(f_x,f_y)}\right\}. \tag{2.8}$$

This however would require the OTF $H(f_x, f_y)$ to be nonzero over all frequencies, and for the image $\hat{I}$ to contain no noise. If such was the case, the division would amplify the noise and small errors, resulting in extreme artifacts. To overcome this, a Wiener filter could be applied [46], which aims to regularize any inversion by weighting each frequency according to the signal-to-noise ratio, restoring attenuated components while preventing excessive amplification of noise.

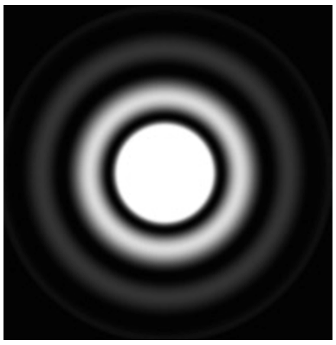


Figure 2.2: Airy disk pattern incident on camera sensor after passing through circular aperture.

In line with Eq. (2.1), to obtain the full OTF of an imaging system, the OTF of the contributing elements must first be obtained. Of these, it is the lens OTF, $H_{lens}(f_x, f_y)$, that is the easiest to model.

When light reaches the camera lens, it first passes through a circular diameter aperture before reaching the camera sensor. The diameter of this aperture D is usually variable, depending on the operator preferences, and can be a deciding factor in image quality. It is more commonly expressed as the ratio $F_{\#} = \frac{f}{D}$, where $F_{\#}$ is the f-number of the camera lens, and f is the camera focal length. The diameter D directly controls the amount of light that is allowed to pass through the lens and onto the camera sensor, therefore determining the sharpness and depth of field (DOF) of the resulting image. For larger values of D, more light is passed through, and produces a shallower DOF, whilst smaller values of D result in a deeper DOF.

This circular aperture causes the incoming light to diffract, resulting in a pattern on the camera sensor known as an Airy disk [47]. This consists of a bright central dot, surrounded by concentric rings of light, with each ring being lighter than the last, as illustrated in Figure 2.2.

The size of this Airy disk is directly controlled by the aperture diameter D and focal length f, defined as,

$$d \approx 1.22\frac{f\lambda}{D} \tag{2.9}$$

where d is the radius from the central bright spot to the first dark ring and λ is the wavelength of light. This Airy disk represents the PSF of the lens due to diffraction, and therefore its OTF, $H_{diff}(f_x, f_y)$.

As for the atmospheric OTF, $H_{atm}(f_x, f_y)$, it is more difficult to model. Due to the nature of image capture, there exists a PSF for each point of light that passes through the atmopshere, i.e. the distortion is spatially varied. When an image is captured, it is the accumulation of these PSFs for each pixel that result in the final image. The distortion due turbulent atmosphere for each of these PSFs can be separated into two operations: warp and blur [29, 33, 48]. In the case of optical turbulence, the warping induced is caused by deviations of the PSF from its centre. This is also known as atmospheric tilt, and is the cause of any 'image dancing' found in the final image. Blur is induced by both the atmosphere OTF, alongside the diffraction OTF, and is directly related to the size of the resulting Airy disk. The full formulation of $H_{atm}(f_x, f_y)$ is detailed in the following sections.

2.2.2 Kolmogorov Theory of Turbulence

The Sun's heating and cooling of Earth's surface lead to large-scale temperature variations in the atmosphere. This, alongside convective air currents, results in fluctuations in the air's refractive index, which is known as *optical turbulence* [42]. It is these fluctuations that alter the path of of optical waves as they travel, resulting in sub-optimal image and video capture. Such a random process however would be impossible to fully simulate, as there are far too many random behaviours and variables to account for. It is for this reason that theories are restricted to statistical analysis. The most widely accepted of these theories is that from Kolmogorov et al. [45]. It is firstly considered that the atmosphere is a viscous fluid that can be categorised as one of two states of motion: laminar or turbulent. In the case of laminar flow, no mixing occurs, resulting in a predictable flow that has known characteristics. The flow becomes turbulent when mixing occurs (i.e small temperature changes and wind), causing the flow to break into subflows called turbulent eddies, where any predictable characteristics are therefore lost [42]. The point at which a fluid can transition from laminar to turbulent flow can be determined by the nondimentional Reynolds number, Re, defined as

$$Re = \frac{Vl}{v}$$

where V, l and v are the velocity, length and kinematic viscosity of the fluid respectively. At low values of the Reynolds number, fluids tend towards laminar flow, whilst high Reynolds numbers characterise turbulent flow [42]. By considering the atmosphere as a fluid, the same principles can be applied in the case of turbulence modelling.

Once a fluid transitions from laminar to turbulent flow, the resulting turbulent

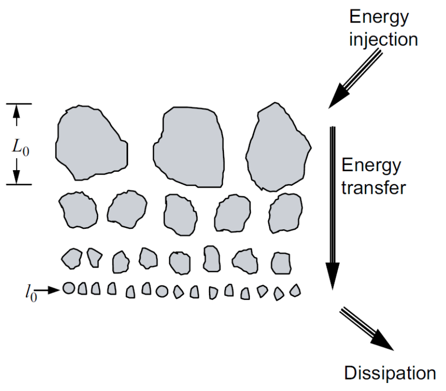


Figure 2.3: Illustration of the energy cascade theory [42].

subflows can then be described as an energy cascade [43]. This begins with large eddies forming due to an injection of energy (either wind or heat); these then proceed to break up into smaller eddies and continue to reduce until the eddies dissipate completely as heat. This cascade begins at a size L_0 and reduces in size to l_0, as illustrated in Figure 2.3. These two values are known as the inner and outer scales of turbulence, where the eddies that lie within these two scales form what is known as the 'inertial subrange' [42].

Within this inertial subrange the eddies are considered to be statistically homogeneous and isotropic. A random function of the spatial vector $\mathbf{R} = (x, y, z)$ is called a random field, denoted as $u(\mathbf{R})$. This field is is considered statistically homogeneous if its mean is independent of the spatial location $\mathbf{R}$, and its covariance is only dependant on the spatial separation $\mathbf{R} = \mathbf{R_2} - \mathbf{R_1}$. Where $\mathbf{R_1}$ and $\mathbf{R_2}$ are two different points in space. It is further considered isotropic if it is also invariant under rotations, i.e. the covariance is dependant only on the *scalar* distance $R = |\mathbf{R_2} - \mathbf{R_1}|$. The use of such mean and covariance values however require the assumption that they are both constant over all space within the field $u(\mathbf{R})$. This is not the case for a turbulent atmosphere however, as the mean will fluctuate over large distances. It is therefore assumed that the random field is *locally homogeneous*, meaning that it can be decomposed as,

$$u(\mathbf{R}) = m(\mathbf{R}) + u_1(\mathbf{R}), \tag{2.10}$$

where $m(\mathbf{R}) = \langle u(\mathbf{R}) \rangle$ is the non-constant mean, and $u_1(\mathbf{R})$ is the fluctuating element with a mean of $\langle u_1(\mathbf{R}) \rangle = 0$. Whilst statistically homogeneous random fields can be well defined by their covariance function, *locally* homogeneous fields are better described by a *structure function*, defined as

$$D_u\left(\mathbf{R}_1, \mathbf{R}_2\right) \equiv D_u(\mathbf{R}) = \left\langle \left[u\left(\mathbf{R}_1\right) - u\left(\mathbf{R}_1 + \mathbf{R}\right)\right]^2 \right\rangle \tag{2.11}$$

which can be expanded as,

$$D_u\left(\mathbf{R}\right) = \left[m\left(\mathbf{R}_1\right) - m\left(\mathbf{R}_1 - \mathbf{R}\right)\right]^2 + \left\langle \left[u_1\left(\mathbf{R}_1\right) - u_1\left(\mathbf{R}_1 - \mathbf{R}\right)\right]^2 \right\rangle, \tag{2.12}$$

where, if the two means $m\left(\mathbf{R}_1\right)$ and $m\left(\mathbf{R}_1 - \mathbf{R}\right)$ are approximately the same, the above can be simplified to,

$$D_u\left(\mathbf{R}\right) \approx \left\langle \left[u_1\left(\mathbf{R}_1\right) - u_1\left(\mathbf{R}_1 - \mathbf{R}\right)\right]^2 \right\rangle. \tag{2.13}$$

Using dimensional analysis, Kolmogorov determined that, within the inertial subrange, the structure function of wind velocity satisfies the universal 2/3 power law [42],

$$D_v\left(R\right) = \left\langle \left(V_1 - V_2\right)^2 \right\rangle = C_v^2 R^{\frac{2}{3}}, \tag{2.14}$$

where V_1 and V_2 are the velocities at two points separated by the distance $\mathbf{R}$, and C_v^2 is the velocity structure parameter, and is a measure of energy within the turbulence. Similar analysis can be performed for the *potential* temperature, providing,

$$D_\theta(r) = C_\theta^2 r^{2/3} \tag{2.15}$$

where C_θ^2 is the potential temperature structure parameter. Note that the *potential* temperature θ is pressure independent, and is linearly related to the absolute temperature T.

As with a locally homogeneous field (Eq. (2.10)), the refractive index at a point $\mathbf{R}$ can be written as,

$$n(\mathbf{R}) = \mu_n(\mathbf{R}) + n_1(\mathbf{R}), \tag{2.16}$$

where $\mu_n(\mathbf{r})$ is the slowly varying mean and $n_1(\mathbf{r})$ is the zero-mean fluctuations. The variations in refractive index is given as [41].

$$dn = 7.99 \times 10^{-5} \frac{d\theta}{T^2}. \tag{2.17}$$

This shows that refractive index variations are directly proportional to the variations in potential temperature, and therefore allows the refractive index structure function to follow the same power law as that of $D_\theta(r)$, giving

$$D_n(R) = C_n^2 R^{2/3}, \tag{2.18}$$

where C_n^2 is the refractive-index structure parameter, and is a measure of refractive index fluctuation strength, measured with units of m$^{-2/3}$ Typical values of C_n^2 range from 1×10^{-16} (weak) to 1×10^{-13} (strong).

The covariance function for a three dimensional statistically homogeneous and isotropic random field is defined as

$$B_n(R) = \frac{4\pi}{R} \int_0^\infty \kappa \Phi_n(\kappa) \sin(\kappa R) d\kappa \tag{2.19}$$

where $\kappa = |\mathbf{K}|$ is the scalar wave number and $\Phi_n(\kappa)$ is the corresponding power spectral density (PSD). The structure function D_n^2 is related to this covariance function by,

$$D_n(R) = 2\left[B_n(0) - B_n(R)\right]. \tag{2.20}$$

This can therefore be expressed as,

$$D_n(R) = 8\pi \int_0^\infty \kappa^2 \Phi_n(\kappa) \left(1 - \frac{\sin \kappa R}{\kappa R}\right) d\kappa \tag{2.21}$$

which, when rearranged for $\Phi_n(\kappa)$, provides [42],

$$\Phi_n(\kappa) = \frac{1}{4\pi^2\kappa^2} \int_0^\infty \frac{\sin \kappa R}{\kappa R} \frac{d}{dR} \left[R^2 \frac{d}{dR} D_n(R)\right] dR. \tag{2.22}$$

Using this and Eq. (2.18), Kolmogorov calculated the *Kolmogorov power-law spectrum*, defined as [45],

$$\Phi_n^K(\kappa) = 0.033 C_n^2 \kappa^{-11/3}, \quad 1/L_0 \ll \kappa \ll 1/l_0 \tag{2.23}$$

where κ is the angular spatial frequency in rad/m. In order for this to be theoretically valid, it is assumed that the outer scale (L_0) is infinite, and the inner scale (l_0) is zero. This introduces restrictions to its use if the inner/outer scales cannot be ignored. It is for this reason that the development of more sophisticated refractive-index PSD was required. Such PSDs were designed such that the inner and outer scales effects could be included, one the simplest of which is the *modified von Kármán* PSD [41], defined as,

$$\Phi_n^{mvK}(\kappa) = 0.033 C_n^2 \frac{\exp\left(-\kappa^2/\kappa_m^2\right)}{\left(\kappa^2+\kappa_0^2\right)^{11/6}} \tag{2.24}$$

where κ is the angular spatial frequency measured in rad/m, $\kappa_m = 5.92/l_0$ and $\kappa_0 = 2\pi/L_0$. Noting that when $l_0 = 0$ and $L_0 = \infty$, Eq. (2.24) reduces to Eq. (2.23).

It is this representation of the Power Spectral Density for turbulent fluctuations that can be utilised to accurately model atmospheric conditions. As will be detailed in Section 2.4, the *modified von Kármán* PSD is used to generate turbulent phase screens, by providing the power at each spatial frequency. It is the propagation through these phase screens that allows the simulation to estimate the optical OTF $H_{atm}(f_x, f_y)$.

2.2.3 Characteristics of Turbulence

Whilst C_n^2 is the most informative parameter for atmospheric turbulence, and is the measure of turbulence strength, other parameters can be defined that allow more control over atmospheric simulation.

One such parameter is the *atmospheric coherence diameter* (or Fried parameter) r_0, which is a measure of optical transmission quality through a turbulent atmosphere [49] measured in meters, with small values of r_0 indicating a more turbulent atmosphere. As stated in Eq. (2.9), the size of the diffraction PSF is controlled by the aperture diameter D. The ratio of this diameter and the Fried parameter, $\frac{D}{r_0}$, is a key indicator when considering the effects of a turbulent atmosphere on an imaging system [50]. In the case that $\frac{D}{r_0} < 1$, the effects of diffraction will outweigh those of turbulence. However, if $\frac{D}{r_0} > 1$, the distortions due to turbulence begin to contribute more towards overall image quality where, for the range of $1 < \frac{D}{r_0} < 2$, the turbulence is considered as weak, and any ratio of $\frac{D}{r_0} > 4$ is considered to be strong turbulence [51]. For a spherical wave, the Fried parameter is defined as,

$$r_{0,sw} = \left[0.423 k^2 \int_0^{\Delta z} C_n^2(z) \left(\frac{z}{\Delta z}\right)^{5/3} dz\right]^{-3/5}, \tag{2.25}$$

which allows the *modified von Kármán* PSD to be redefined in terms of r_0 as [41],

$$\Phi_{\phi_i}^{mvK}(f) = \frac{0.023 \mathrm{e}^{-f^2/f_m^2}}{r_0^{5/3}\left(f^2+f_0^2\right)^{11/6}}, \tag{2.26}$$

where the PSD is now written in terms of f, which is the frequency measured in cycles/m, $f_m = 5.92/2\pi l_0$ and $f_0 = 1/L_0$.

Another key parameter of atmospheric turbulence is the isoplanatic angle θ_0, which describes the extent of anisoplanatism within the simulation and is measured in radians. A system is considered anisoplanatic if the OTF, and resulting PSF, are not shift-invariant [41], i.e. the angular path through the system will change the resulting PSF. If two point sources are separated by more than this angle, their path through the atmosphere is assumed to be different, resulting in different PSFs. For large values of θ_0, the system tends towards isoplanatism, i.e all paths through the atmosphere result in the same OTF. The isoplanatic angle is defined as,

$$\theta_0 = \left[2.91k^2\Delta z^{5/3}\int_0^{\Delta z} C_n^2(z)\left(1-\frac{z}{\Delta z}\right)^{5/3} dz\right]^{-3/5} \tag{2.27}$$

The final atmospheric parameter to define is that of the log-amplitude variance σ_x^2. This is a statistic that represents the fluctuations in the amplitude of the wave as it is propagated [52]. High values of σ_x^2 relate to a higher variance, therefore indicating more severe fluctuations in the wave amplitude. It is defined as,

$$\sigma_{\chi,sw}^2 = 0.563k^{7/6}\int_0^{\Delta z} C_n^2(z)z^{5/6}\left(1-\frac{z}{\Delta z}\right)^{5/6} dz. \tag{2.28}$$

Together, these three parameters allow the *modified von Kármán* PSD to be accurately calculated at any point along the propagation path of the simulation. Without these parameters, various assumptions would need to be made regarding the characteristics of the atmosphere, and would lead to a inferior simulation.

2.2.4 Split-Step Propagation

The propagation of a light source through a turbulent medium is achieved through the split-step beam propagation method [41]. This method facilitates the propagation of a wave by separating the full propagation distance into discrete layers. This discretisation of the full volume allows an assumption to be made that the statistics of the atmosphere within each layer are approximately constant [5]. The full propagation can then be achieved by simply performing a propagation through each layer in turn where, for each layer, two effects can be defined: diffraction and refraction [41]. Diffraction is the effect caused by free space propagation from one point to another, whilst refraction is caused by the propagation medium itself, i.e. the atmosphere, and is is modelled by a unit-amplitude thin

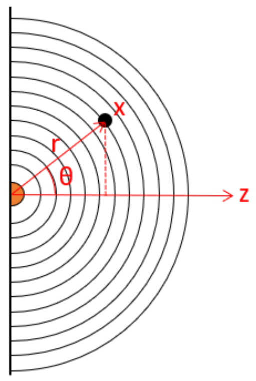


Figure 2.4: Illustration of a single point emitting a spherical wave, as per the Huygens-Fresnel principle.

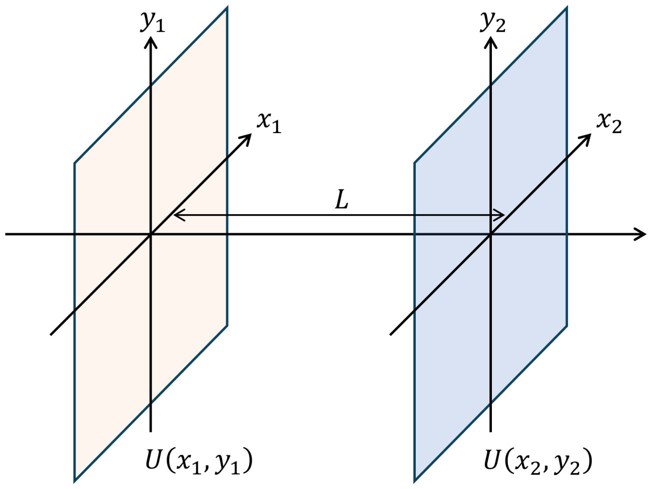


Figure 2.5: Coordinate system of two planes. Fresnel diffraction allows propagation from one plane $U(x_1, y_1)$ to another $U(x_2, y_2)$.

phase screen, which represents the phase perturbation of that particular sub-volume. This is valid as long as the atmosphere within the layer is considered to be statistically homogeneous and isotropic [41].

The Huygens-Fresnel principle [53] states that any point on a wavefront can be considered as a new source of a spherical wave [54], i.e. it can be assumed that each point on a plane emits its own spherical wave. This is illustrated in Figure 2.4, where the wave propagates more strongly along the z direction, and attenuates as it travels. The amplitude of this wave at any point x can found, in complex notation, as

$$U(x,t) = \frac{z}{r}\frac{e^{ikr}}{r} = \frac{z}{r^2}e^{ikr}, \tag{2.29}$$

where e^{ikr} describes the spherical field [54], $\frac{z}{r}$ is the off-axis attenuation factor and remaining $\frac{1}{r}$ is the distance attenuation factor.

Figure 2.5 illustrates two planes, separated by a distance $z = L$. As per the Huygens-Fresnel principle, every point on the source plane $U(x_1, y_1)$ acts as a source emitting a spherical wave. Consequently, the field at any point on the observation plane $U(x_2, y_2)$ can be found as the superposition of contributions from all points on $U(x_1, y_1)$. The total field $U(x_2, y_2)$ is therefore obtained by integrating the effects of all the sources across the source plane. Such an integration is also known as the Rayleigh-Sommerfeld Integral [54], and is given as,

$$U(x_2, y_2) = \frac{1}{i\lambda} \iint U\left(x_1, y_1\right) \frac{L}{r^2} e^{ikr} \, \mathrm{d}x_1 \mathrm{d}y_1 \tag{2.30}$$

where r denotes the distance between two points on $U(x_1, y_1)$ and $U(x_2, y_2)$, calculated using the Pythagorean theorem as,

$$r = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2 + L^2}. \tag{2.31}$$

The factor of $\frac{1}{r^2}$ in Eq. (2.30) can be approximated as $\frac{1}{L^2}$, assuming that $L^2 >> \left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2$, giving,

$$U(x_2, y_2) = \frac{1}{i\lambda L} \iint U\left(x_1, y_1\right) e^{ikr} \, \mathrm{d}x_1 \mathrm{d}y_1 \tag{2.32}$$

The same assumption cannot however be made in e^{ikr}, therefore requiring a more careful approach. Eq. (2.31) can be expanded as

$$r = L\sqrt{1 + \frac{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}{L^2}} \tag{2.33}$$

where, with the assumption that $L^2 >> \left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2$, the square root can be approximated with a first order Taylor expansion ($\sqrt{1+\alpha} \approx 1 + \frac{\alpha}{2} - \dots$), giving,

$$\begin{aligned} r &\approx L\left(1 + \frac{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}{2L^2}\right) \\ &\approx L + \frac{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}{2L}. \end{aligned} \tag{2.34}$$

This is the Fresnel approximation, and allows Eq. (2.32) to be rewritten as the

Fresnel diffraction equation, defined as, [41]

$$\begin{aligned} U\left(x_2, y_2\right) =& \frac{e^{ikL}}{i\lambda L} e^{ik\frac{\left({x_2}^2+{y_2}^2\right)}{2L}} \\ &\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} U\left(x_1, y_1\right) e^{ik\frac{\left({x_1}^2+{y_1}^2\right)}{2L}} e^{-2\pi i\left(x_2\frac{x_1}{\lambda L}+y_2\frac{y_1}{\lambda L}\right)} dx_1 dy_1. \end{aligned} \tag{2.35}$$

By comparing this with Eq. (2.2), it can be observed that the exponent within the Fresnel diffraction equation is the same as that found in the Fourier transform, allowing the Fresnel diffraction equation to be rewritten as,

$$\mathrm{U}(x_2, y_2) = \frac{e^{ikL}}{i\lambda L} e^{ik\frac{\left({x_2}^2+{y_2}^2\right)}{2L}} \mathcal{F}\left\{\mathrm{U}(x_1, y_1) e^{ik\frac{\left({x_1}^2+{y_1}^2\right)}{2L}}\right\}\left(\frac{x_1}{\lambda L}, \frac{y_1}{\lambda L}\right) \tag{2.36}$$

In order to expand this equation to allow multiple propagations, as per the split-step propagation method, it is first useful to define spatial $(\boldsymbol{r_1}, \boldsymbol{r_2})$ and spatial-frequency vectors $(\boldsymbol{f_1})$ as,

$$\boldsymbol{r_1} = (x_1, y_1) \tag{2.37}$$

$$\boldsymbol{r_2} = (x_2, y_2) \tag{2.38}$$

$$\boldsymbol{f_1} = (f_{x1}, f_{y1}) \tag{2.39}$$

It is also useful to introduce operator notation, which is executed from right to left and puts emphasis on the operations taking place. Such operations are defined as,

$$Q[c, \boldsymbol{r}]\{U(\boldsymbol{r})\} = e^{i\frac{k}{2}c|\boldsymbol{r}|^2} U(\boldsymbol{r}) \tag{2.40}$$

$$V[b, \boldsymbol{r}]\{U(\boldsymbol{r})\} = bU(b\boldsymbol{r}) \tag{2.41}$$

$$\mathcal{F}[\boldsymbol{r}, \boldsymbol{f}]\{U(\boldsymbol{r})\} = \int_{-\infty}^{\infty} U(\boldsymbol{r}) e^{-i2\pi \boldsymbol{f}\cdot\boldsymbol{r}} d\boldsymbol{r} \tag{2.42}$$

$$\mathcal{F}^{-1}[\boldsymbol{f}, \boldsymbol{r}]\{U(\boldsymbol{f})\} = \int_{-\infty}^{\infty} U(\boldsymbol{f}) e^{i2\pi \boldsymbol{f}\cdot\boldsymbol{r}} d\boldsymbol{f} \tag{2.43}$$

which allows Eq. (2.36) to be rewritten again as [41],

$$U\left(r_2\right) = Q\left[\frac{1}{L}, \boldsymbol{r}_2\right] V\left[\frac{1}{\lambda L}, \boldsymbol{r}_2\right] \mathcal{F}\left[r_1, f_1\right] Q\left[\frac{1}{L}, \boldsymbol{r}_2\right]\left\{U\left(\boldsymbol{r}_1\right)\right\}, \tag{2.44}$$

where the phase factor e^{ikL} in Eq. (2.36) can be ignored when using a reduced field distribution [55]. It is at this point that the Fresnel diffraction equation can be expanded for multiple consecutive propagations, and facilitate multi-step propagation. The following is the final form of this expansion

$$\begin{aligned}U\left(\boldsymbol{r}_n\right) =&Q\left[\frac{m_{n-1}-1}{m_{n-1}\Delta z_{n-1}},\boldsymbol{r}_n\right]\\&\prod_{i=1}^{n-1}\left\{\mathcal{F}^{-1}\left[\boldsymbol{f}_i,\frac{\boldsymbol{r}_{i+1}}{m_i}\right]Q_2\left[-\frac{\Delta z_i}{m_i},\boldsymbol{f}_i\right]\mathcal{F}\left[\boldsymbol{r}_i,\boldsymbol{f}_i\right]\left(\frac{1}{m_i}\right)\right\}\\&\left\{Q\left[\frac{1-m_1}{\Delta z},\boldsymbol{r}_1\right]U\left(\boldsymbol{r}_1\right)\right\},\end{aligned} \tag{2.45}$$

where the key changes made between Eq.s (2.44) and (2.45) include the expansion to allow different grid sizes between planes ($m = \frac{\Delta_2}{\Delta_1}$) and the generalisation of up to n partial propagations [41]. Using this final form of the Fresnel diffraction equation, any point source $U(\mathbf{r}_1)$ can be propagated through the atmosphere to provide the resulting field at the observation plane $U(\mathbf{r}_n)$. This final plane represents how the point source has changed as it has propagated through the atmosphere, and therefore represents the output of the OTF for that particular path. It is from this that the final PSF for each light source can be evaluated.

2.2.5 Formulation of Point Source

In an ideal scenario, the point source used for propagation would be a Dirac delta function. This however has a constant value across all spatial frequencies, and therefore has an infinite spatial bandwidth. Due to the sampling restraints of a digital simulation, this is therefore not feasable. To overcome this, a bandlimited point source must be used. This is achieved by first representing the point source as $\widetilde{U}_{pt}\left(\boldsymbol{r}_1\right)$ and substituting it into the the Fresnel diffraction equation (Eq. (2.35)) as,

$$\widetilde{U}\left(\boldsymbol{r}_2\right) = \frac{e^{ikL}}{i\lambda L}e^{i\frac{k}{2L}\left(r_2^2\right)}\int_{-\infty}^{\infty}\widetilde{U}_{pt}\left(\boldsymbol{r}_1\right)e^{i\frac{k}{2L}\left(r_1^2\right)}e^{-i\frac{2\pi}{\lambda L}\left(\boldsymbol{r}_1\boldsymbol{r}_2\right)}d\boldsymbol{r}_1$$

which, after simplification and rearranging for $\widetilde{U}_{pt}\left(\boldsymbol{r}_1\right)$ provides,

$$\widetilde{U}_{pt}\left(\boldsymbol{r}_1\right) = i\lambda Le^{ikL}e^{-i\frac{k}{2L}\left(r_1^2\right)^2}\mathcal{F}^{-1}\left\{\widetilde{U}\left(\lambda L\boldsymbol{f}_1\right)e^{-i\pi\lambda Lf_1^2}\right\} \tag{2.46}$$

where $\widetilde{U}\left(\lambda L\boldsymbol{f}_1\right)$ is the complex field at the observation plane. This can then further be simplified by defining the desired field at the observation plane as,

$$\widetilde{U}\left(\boldsymbol{r}_2\right) = \frac{e^{ikL}}{i\lambda L}W\left(\boldsymbol{r}_2-\boldsymbol{r}_c\right)e^{i\frac{k}{2L}\left|\boldsymbol{r}_2-\boldsymbol{r}_c\right|^2}, \tag{2.47}$$

where $\mathbf{r}_2 = x_2\hat{\mathbf{i}} + y_2\hat{\mathbf{j}}$ is the spatial vector in the observation plane, $\mathbf{r}_c = (x_c, y_c)$

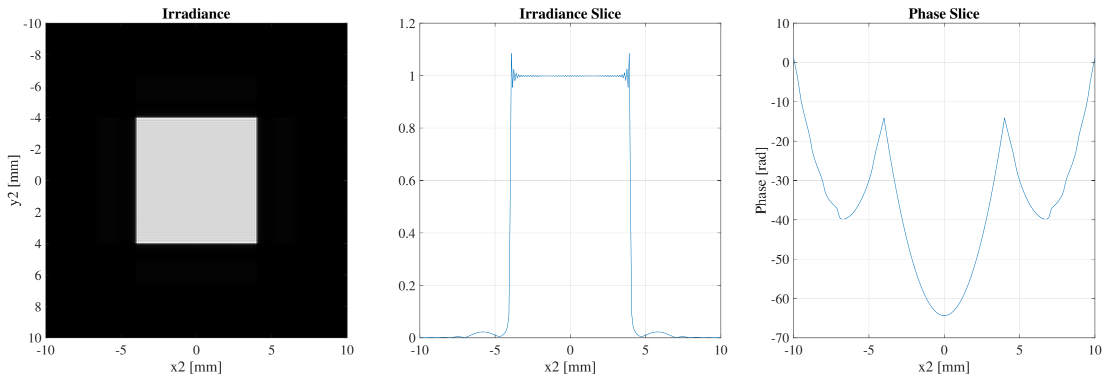


Figure 2.6: Propagated irradiance and phase of the point source defined in Eq. (2.50) for a region with a width of $D = 8mm$.

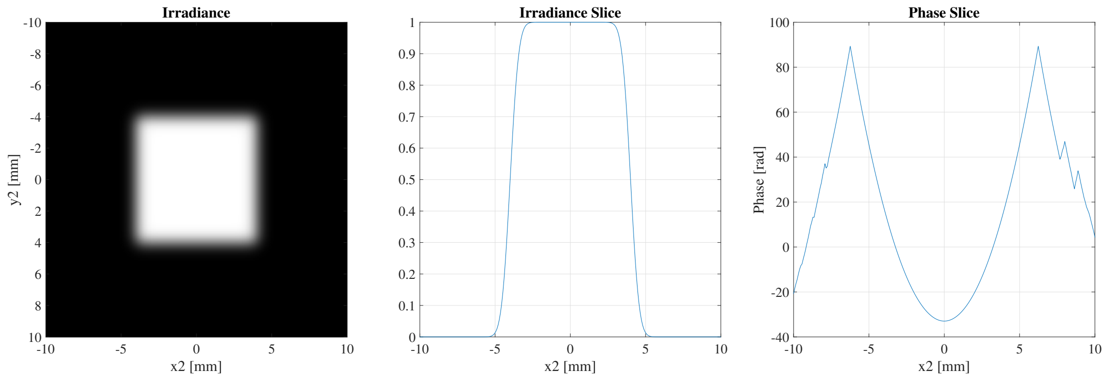


Figure 2.7: Propagated irradiance and phase of the gaussian point source defined in Eq. (2.51) for a region with a width of $\widetilde{D} = 8mm$.

is the location of the point source in the source plane and $W(\mathbf{r}_2 - \mathbf{r}_c)$ represents the aperture of the camera. Eq. (2.46) can therefore be rewritten as,

$$\widetilde{U}_{pt}\left(\boldsymbol{r}_1\right) = e^{-i\frac{k}{2L}\left(r_1^2\right)} e^{i\frac{k}{2L}\left(r_c^2\right)}{}_{\mathcal{F}}\mathcal{F}^{-1}\left\{W\left(\lambda L \boldsymbol{f}_1 - \boldsymbol{r}_c\right) e^{-i2\pi \boldsymbol{r}_c \boldsymbol{f}_1}\right\}. \tag{2.48}$$

The operator W can be defined as a square region with a width of D as,

$$W\left(\boldsymbol{r}_2 - \boldsymbol{r}_c\right) = A\,\mathrm{rect}\left(\frac{x_2 - x_c}{D}\right)\mathrm{rect}\left(\frac{y_2 - y_c}{D}\right) \tag{2.49}$$

which, when substituted into Eq.(2.48), simplifies to [41],

$$\widetilde{U}_{pt}\left(\boldsymbol{r}_1\right) = \lambda L e^{-i\frac{k}{2L}\left(r_1^2\right)}\left(\frac{D}{\lambda L}\right)^2 \mathrm{sinc}\left[\frac{D\left(x_1 - x_c\right)}{\lambda L}\right] \sin c\left[\frac{D\left(y_1 - y_c\right)}{\lambda L}\right] e^{-\left(\frac{D}{4\lambda L r_1}\right)^2} \tag{2.50}$$

Figure 2.6 shows the resulting irradiance field after propagation of Eq. (2.50). It can be seen that the resulting field is as desired, i.e. square region with width D. However, analysis of the phase shows aliasing outside the region of interest.

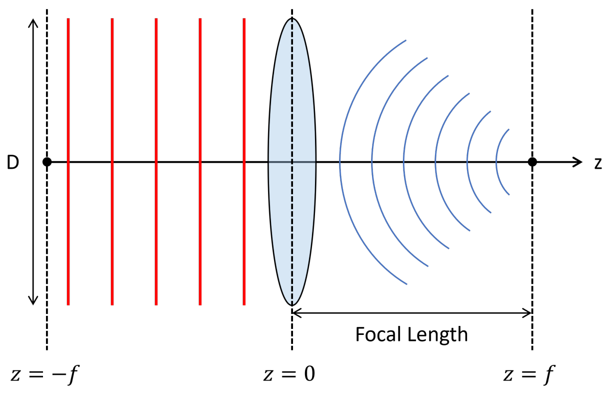


Figure 2.8: Illustration of a lens transform, that converts a plane wave into a spherical wave focused at the focal length f.

It is therefore desirable to mitigate this aliasing by introducing a Gaussian into the point source equation. Such a gaussian point source can be defined as,

$$U_{pt}(x,y) = \lambda L \alpha^2 e^{-i\frac{k}{2L}\left(x^2+y^2\right)} \operatorname{sinc}[\alpha x, \alpha y] e^{-\frac{\alpha^2}{16}\left(x^2+y^2\right)} \tag{2.51}$$

where $\alpha = \frac{\widetilde{D}}{\lambda L}$ and $\widetilde{D}$ is the desired width of the amplitude field at the observation plane, as illustrated in Figure 2.7 , where it can be seen that any aliasing has successfully been minimised by the included Gaussian.

2.2.6 Lens Transfer Function

The lens of a camera can be represented mathematically by first showing that a planar wave incident on a camera lens is converted to a spherical wave that converges to the focal point, as illustrated in Figure 2.8. The transfer function of such a lens is defined as,

$$e^{i\phi lens} = e^{-ik\frac{x^2+y^2}{2f}}. \tag{2.52}$$

Therefore, to obtain the field at the back of the lens, the wave at the front of the lens, $\mathrm{U}(x,y,0)$, is simply multiplied by this transfer function [54] as,

$$\mathrm{U}(x,y,0)e^{-ik\frac{\left(x^2+y^2\right)}{2f}}$$

This spherical wave can then be propagated to the focal plane with the Fresnel diffraction equation as defined in Eq. (2.36) giving,

$$\begin{aligned}\mathrm{U}(x,y,f) &= e^{ik\frac{\left(x^2+y^2\right)}{2f}}\mathcal{F}\left\{\mathrm{U}(x,y,0)e^{-ik\frac{\left(x^2+y^2\right)}{2f}}e^{ik\frac{\left(x^2+y^2\right)}{2f}}\right\}\left(\frac{x}{\lambda f},\frac{y}{\lambda f}\right)\\ &= e^{ik\frac{\left(x^2+y^2\right)}{2f}}\mathcal{F}\{\mathrm{U}(x,y,0)\}\left(\frac{x}{\lambda f},\frac{y}{\lambda f}\right).\end{aligned} \tag{2.53}$$

where $\mathrm{U}(x,y,f)$ is the field at the focal length and it can again be seen that a constant phase factor has been dropped [54].

Whilst Eq. (2.53) would successfully obtain the field at the focal plane, the factor of $e^{ik\frac{\left({x_2}^2+{y_2}^2\right)}{2f}}$ is cumbersome and can be removed by analysing the relationship between the field at the front focal plane, $\mathrm{U}(x,y,-f)$, and $\mathrm{U}(x,y,0)$. The relationship between these two planes is defined by the angular-spectrum propagation of a wave [41] as,

$$\mathcal{F}\{\mathrm{U}(x,y,0)\}\left(\frac{x}{\lambda f},\frac{y}{\lambda f}\right)\approx\mathcal{F}\{\mathrm{U}(x,y,-f)\}\left(\frac{x}{\lambda f},\frac{y}{\lambda f}\right)e^{-ik\frac{\left(x^2+y^2\right)}{2f}} \tag{2.54}$$

which can be substituted into Eq. (2.53) to give

$$\mathrm{U}(x,y,f)=e^{ik\frac{\left({x_2}^2+{y_2}^2\right)}{2f}}\mathcal{F}\{\mathrm{U}(x,y,-f)\}\left(\frac{x}{\lambda f},\frac{y}{\lambda f}\right)e^{-ik\frac{\left({x_2}^2+{y_2}^2\right)}{2f}} \tag{2.55}$$

where the two quadratic phase factors cancel out to give,

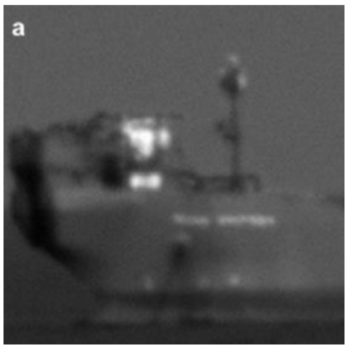
$$\mathrm{U}(x,y,f)=\mathcal{F}\{\mathrm{U}(x,y,-f)\}\left(\frac{x}{\lambda f},\frac{y}{\lambda f}\right). \tag{2.56}$$

Given this, an assumption can be made that the observed field after propagation lies at $z = L - f$ instead of $z = L$. This allows the field at $z = L + f$ to be obtained via a simple Fourier transform.

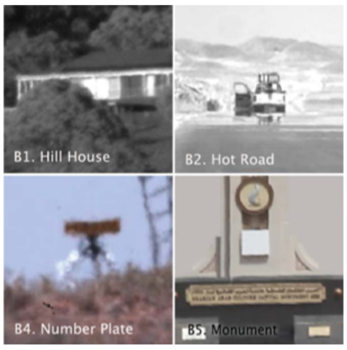
The final step required before an PSF is obtained is that of conversion from field amplitude to intensity. For incoherent light, this is achieved via a squaring operation [41] as,

$$\mathrm{I}(x,y,f)=|\mathrm{U}(x,y,f)|^2 \tag{2.57}$$

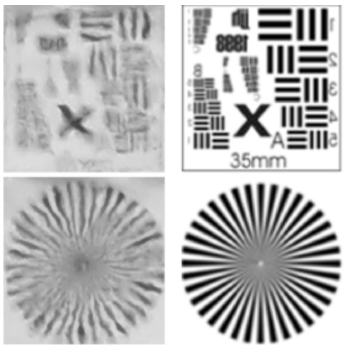
This therefore represents the final PSF of the turbulent path through the atmosphere, and can be applied to the final image. By repeating this process for each source of light in an image, a final turbulence image can be formulated.



(a) Turbulent image of a ship from Nieuwenhuizen et al. [56].



(b) Turbulence examples from CLEAR dataset [57].



(c) Turbulent and ground truth resolution pattern from OTIS dataset [58].

Figure 2.9: Examples of real turbulence images found in literature.

2.3 Real Life Turbulence Datasets

Suitable data is paramount for the development of turbulence mitigation algorithms. The distortion due to turbulence is extremely varied, and can change due to a multitude of different factors such as wind velocity, heat and object distance. The acquisition of suitable data is therefore an extremely difficult task as, if the data used for development does not match that during validation or testing, the algorithm will perform poorly. Not only this, but for algorithms such as deep learning, a turbulence-free image is also required for each distorted one, which is impossible to obtain in real applications without direct control over the atmosphere itself.

Given these difficulties however, real data is still highly valuable for validation and testing. Simulation techniques, for example, require the knowledge that the synthesised images are as realistic as possible. Therefore, by comparing with real data, they can be validated. As for turbulence mitigation algorithms, the removal of turbulence from synthetic data is pointless unless that same algorithm can be used in practice. Real data is therefore useful for testing such algorithms on realistic distortions. In order to supply such data, various sources have either acquired their own data for internal validation, or acquired data for public use, examples of which can be seen in Figure 2.9.

One such dataset for turbulence related processing is the Open Turbulent Image Set (OTIS) [58], which contains various video sequences of both static and dynamic scenes, where static refers to a scene with a fixed camera and no object movement, and dynamic refers to scenes where the camera, or objects in the scene, may move. The static scenes consist of a known pattern printed onto a piece of paper, which can then be used as the ground truth. For the dynamic scenes, the videos consist of a remote controlled car being driven through the scene; for these, no turbulence-free ground truth images are available, however the location of the car within each frame is available for tracking purposes.

Anantrasirichai et al. have twice collated real data for the purpose of turbulence mitigation: CLEAR (Complex waveLEt fusion for Atmospheric tuRbulence) [57] and CLEAR2 [13]. In the case of CLEAR, six dynamic sequences were obtained with real turbulence distortion present, and for CLEAR2, seven additional sequences were obtained via direct capture or were supplied by external sources. For each of these sequences, no ground truth is present, but are mostly publicly available. Similarly, Jin et al. [59] were able to capture a large dataset of turbulence distorted data in Australia, again without ground truth data, and is also publicly available.

In 2018, the Computer Vision and Pattern Recognition (CVPR) conference held the UG^2 (Now UG^2+) competition for image enhancement, with the specific goal of facilitating the automation of post-acquisition processing. In 2022, this competition expanded to include a specific track for atmospheric turbulence mitigation. For this challenge task a new dataset was acquired for testing purposes; the Turbulence Text Dataset [60]. It comprises of videos displaying printed text on paper, captured through a real turbulent atmosphere over 300m.

More recently, Xu et al. [61] presented a dataset named RLR-AT. This dataset contains 1500 sequences with approximately 800 frames per sequence, taken over distances from 1 to 13km. Whilst the ground truth to these sequences is unavailable, the dataset is publicly available and represents one of the more comprehensive datasets in literature.

Other examples of the acquisition of real turbulent data for mitigation validation can be found from Nieuwenhuizen et al. [56] and Piscaer et al. [24], both of which focus on the processing of videos of ships at sea. In the case of [56], real images of ships are acquired over a varied distance between 5 and 10km, and the performance of the mitigation algorithms are assessed visually, as no ground truth was possible. The work from Piscaer et al. focuses on the recognition of text *through* the turbulent atmosphere, as opposed to mitigation. In this, they captured videos of ships at distances of 10 to 15km, and were able to obtain the ground truth ship names from publically available sources. Neither of these works have made their data publicly available.

Obtaining turbulent images/videos is vital for turbulence mitigation algorithms, however it may not always be the clean image that is required. In some cases, it is more beneficial to understand the level of turbulence within a given sequence, i.e. what is the representative C_n^2 value. In these cases, it is not just the turbulent images/videos that is required, but also the ground truth C_n^2 value, which can be obtained with a scintillometer. One such dataset, that is publicly available, is from Saha et al. [62], who make use of their labelled data to train a deep learning architecture to perform C_n^2 estimation. Similarly, Tunick et al. [8] took C_n^2 measurements over a distance of 2.3km for the purposes of turbulence characterisation for different atmospheric conditions. Using this data, a better understanding of how the atmosphere may affect future optical communication systems was obtained. Another example of turbulence characterisation is the False Bay ATMOSpheric Experiment (FATMOSE) trial [48]. In this trial, the C_n^2 was measured for a distance of approx 15.7km, and was measured over the span of a year in order to capture seasonal variations. Using this data, the authors presented a methodology for estimating the PSF by examining the size of point sources in the captured images.

Turbulence mitigation algorithms can also be evaluated based on various statistics that can be extracted from an image. It is common for the limitations of camera systems to be found via the use of a resolution chart [30], where a higher quality system will be able to capture higher frequency components. This can also be achieved for turbulence mitigation by acquiring pictures of a resolution chart through the atmosphere. Such was performed by Boehrer et al. [51], who captured three different targets at three different ranges (800m, 1500m & 4000m). Then, from these charts, metrics such as blur and tilt were extracted, therefore allowing the better mitigation algorithm to be identified. Another use of resolution charts can be found in the work from Miller et al. [63] who make use of a known pattern to validate their turbulence simulation method. By capturing the C_n^2 value during image acquisition, the same distortion can be simulated and applied to a clean image. The real and simulated distorted pattern can then be compared.

2.4 Turbulence Simulation

To overcome the difficulties and drawbacks of real turbulent data acquisition, the distortion is most commonly simulated, and applied to clean images. This can be done in two ways, either with physical simulation, or computer simulation. Where computer simulation can be achieved via traditional mathematical models or via deep learning methods.

2.4.1 Physical simulation

Physical simulation is the process of creating a controllable environment, in which the turbulence can be varied, allowing not only a ground truth image to be obtained, but also different levels of turbulence strength. The most commonly cited in literature is that from Hirsch et al. [64] who made use of a heating vent on the top of a building to provide a turbulent atmosphere. By closing this vent, the clean ground truth images could be also be obtained. The three images 'Books', 'Chimney' and 'Building' can be found as common 'real-life' turbulent examples in turbulence mitigation literature [14, 16, 25, 57, 65, 66].

Other methods of physical simulation include hair driers [67], heat plates [68], gas burners [57, 59], and heat chambers [60]. Where in the case of [59] and [60], the clean videos were displayed on a screen, therefore allowing any scene to be 'simulated'. Many of these physical simulations were presented alongside the real-life data described previously [57, 59, 60], where researchers made use of the convenience and control of physical simulation alongside the theoretical accuracy of real-life acquisition.

Whilst these methods introduce an element of control of the 'turbulent' atmosphere, they are not true simulations of turbulence, and commonly rely on heat for their effects. This means that they do not fully capture the complex nature of long range atmospheric turbulence, and therefore imply an imitation of real distortions caused by a turbulent atmosphere. Therefore, to allow the simulation of long distance turbulence, as well as full control over the atmosphere, computer simulations are commonly used.

2.4.2 Computer simulation

When developing computer simulations, a trade-off exists between accuracy and processing time where, in order to ensure an accurate simulation, a large number of calculations are needed, therefore increasing simulation time. Conversely, other simulation methods that make use of turbulence statistics are faster, but are limited in their ability to fully represent the turbulent atmosphere.

As described by Eq. (2.1), any distortion present within an image can be described by the system OTF $H(f_x, f_y)$ which, in the case of turbulence, can be represented by spatially varying warp and blur. It is therefore these two effects that computer simulations aim to recreate.

The simplest way to represent $H(f_x, f_y)$ for turbulence is to first isolate the two effects, and apply them independently, where blur is commonly represented

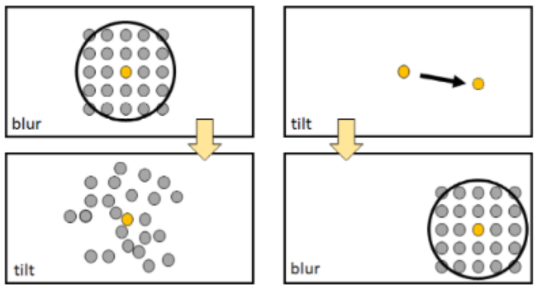


Figure 2.10: Resulting pixel locations of blur-then-tilt (left) and tilt-then-blur (right) models. Noting that the blur-then-tilt model results in the blur structure being lost, and therefore incorrect.

by a Gaussian kernel, and the warping (tilt) function being the focus of most simulation literature. The work from Chan et al. [69] takes this concept and performs an investigation into the order of operations of these two effects, posing the question "Should it be tilt-then-blur or blur-then-tilt?", ultimately showing that the tilt-then-blur model is correct. The key difference lying in the fact that tilt-then-blur models first moves pixels to a new location, and blurs the result. Whereas blur-then-tilt models end up moving the blurred points to new locations, therefore losing the effect of the blur, as illustrated in Figure 2.10.

Examples of tilt-then-blur models can be found in the work by Zhu et al. [70] or Chak et al. [25], who model the spatially varied warp by sampling from a normal distribution, and model the blur with a Gaussian kernel. Whilst this approach is correct in its order of operations, warping the image with random values is not recommended, as it does not capture the full complexity and nature of atmospheric turbulence.

As previously described, the practical acquisition of turbulent data is a challenging task. However, the acquisition of of real data can greatly facilitate the development of accurate computer simulations. A popular method of simulation is from Repasi et al. [71, 72], who captured images of a checkerboard pattern through atmospheric turbulence. By analysing the width and excursion of the edges, they were able to draw valid statistics regarding the pixel movement due to turbulence. This therefore allows their simulation method to draw upon real turbulent experience when synthesising an image. This method has also been used in [56, 65, 73]. Miller et al. [63], then make further improvements upon this method by first replacing the Gaussian blurring kernel with that of the short exposure OTF for atmospheric turbulence.

Using this OTF is also a common method of turbulence simulation. Instead of applying the two effects of warp and blur, a single PSF can be generated that represents both effects. Such PSFs are estimated using the OTF of an imaging system, as defined in Eq. (2.1) where the atmospheric OTF, $H_{\text{atm },\alpha}(\rho)$, is defined as [46],

$$H_{\text{atm},\alpha}(\rho) = \exp\left\{-3.44\left(\frac{\lambda l \rho}{r_0}\right)^{5/3}\left[1-\alpha\left(\frac{\lambda l \rho}{D}\right)^{1/3}\right]\right\} \tag{2.58}$$

where $\rho = \sqrt{u^2+v^2}$, and u and v are the spatial frequency coordinates measured in cycles/m. The variable α is a measure of tilt compensation, with $\alpha = 0$ being no compensation, and $\alpha = 1$ being full compensation. The two resulting transfer functions, $H_{\text{atm},0}(\rho)$ and $H_{\text{atm},1}(\rho)$, relate to the long exposure (LE) and short exposure (SE) OTFs respectively. The diffraction-limited OTF for a circular pupil, $H_{\text{dif}}(\rho)$, is defined as [74],

$$H_{\text{dif}}(\rho) = \begin{cases} \frac{2}{\pi}\left[\cos^{-1}\left(\frac{\rho}{\rho_c}\right) - \frac{\rho}{2\rho_c}\sqrt{1-\left(\frac{\rho}{\rho_c}\right)^2}\right] & \rho \le \rho_c \\ 0 & \text{else,} \end{cases} \tag{2.59}$$

where ρ_c is the optical cut-off frequency and is defined as

$$\rho_c = \frac{D}{\lambda l}.$$

By modelling the atmosphere with $H_{\text{atm},\alpha}$ and H_{dif}, a PSF for the entire imaging system can be found via Eq. (2.3). Such approaches to turbulence simulation can be found in [33, 75], as well in the work by Gao et al. [66], who make use of previously generated PSFs.

By representing the Kolmogorov PSD with Zernike polynomials, Chimitt et al. [49] developed an accurate method of turbulence simulation which is able to generate spatially correlated PSFs for each pixel within an image. This was further improved upon by Mao et al. [76], who developed a shallow deep learning network that allows the Zernike coefficients to be transformed into PSFs basis coefficients, removing the costly operation of per pixel Fourier transforms and spatially varied convolution. Even further improvements to this simulation are made by Chimitt et al. in [77], where the pre-computation of the cross-correlation tensor is replaced with a memory efficient approximation method. Furthermore, the simulator is expanded to allow a near-continuous C_n^2 profile in [78].

Whilst 2D simulations can provide an indication of a turbulent volume, the most accurate method of depicting the path of light is by implementing a propagation simulation [79]. Such a propagation for a single light source can be achieved with minimal computation; however the aggregate time required to simulate the propagation for each pixel can result in a computationally expensive simulation. Given this however, such a simulation allows a greater level of control over the atmospheric affects, and results in a more accurate model. Such models represent the atmosphere with a series of complex planes called phase screens [16, 80], which are generated from the modified von Kármán PSD as defined in Eq. (2.26). The availability of such simulations is extremely limited, as most are developed for internal use only [81–83]. However, sources such as 'Numerical Simulation of Optical Wave Propagation' from Jason D.Shmidt [41] or the python package 'wavepy' from Beck et al. [80] provide useful tools for the development of such a simulator.

2.4.3 Deep learning based simulation

The final method of turbulence simulation is the use of deep learning. Whilst more details of deep learning and its various applications are discussed in Chapter 3, these methods are briefly introduced here for reference. The work from Rai et al. [84] make use of a Generative Adversarial Network (GAN) to generate new turbulent samples. They make use of the simulator described by Schwartzman et al. [85] to generate a dataset with which to train their model. Franz et al. [86] on the other hand aim to generate an array of PSFs instead of the turbulent image directly. They achieve this by training a conditional Variational Autoencoder (cVAE) with real turbulent images of an LED (Light Emitting Diode) array. As the images are taken through atmospheric turbulence, the effective PSF is captured for each LED.

Once trained, a deep learning approach to turbulence simulation represents the fastest possible method, as the inference time does not require complex calculations and can be greatly optimised on appropriate hardware (GPUs). However, due to the fact that the models are trained with a limited distribution of turbulent images, the resulting models will only be able to generate samples within that same distribution. Therefore, whilst beneficial in generating more of what is already available, their ability to generate turbulence outwith their training data is a significant disadvantage.

Due to the ever growing popularity of deep learning approaches to computer vision tasks, the creation and curation of dedicated datasets has become extremely important. It is for this reason that many of the simulation methods described (physical and computer) have synthesised data that is now publicly available. Of the physical simulations, the data from Anantrasirichai at al. [57] and Hirsch et al. [64] are publicly available, or are available with permission from the authors. The work from Jin et al. [59] produced a variety of data, included physical simulation (gas burners), computer simulation (PSF generation) and real life data, all of which have are open source. Throughout its various stages of development, the simulator from Chimitt et al. [49] has been utilised to synthesise a number of useful datasets, where the work from Mao et al. [76] is the core simulation method used for the UG^2+ competition. Furthermore, in the two works from Zhang et al. [87, 88], four different datasets were introduced. These were Turbsyn-static and TurbSyn-dynamic from [87] and ATSyn-static and ATsyn-dynamic from [88]. By separating the data into static and dynamic movement, the specific development of algorithms in each domain is facilitated.

2.5 Traditional Turbulence Mitigation

The tasks of turbulence simulation and turbulence mitigation are linked by the formulation in Eq. (2.1), where simulation methods aim to represent $H(f_x, f_y)$ for turbulence distortion. For turbulence mitigation, it is the inverse of $H(f_x, f_y)$ that is required, such that the original clean image can be recovered.

The difficulty in turbulence mitigation is in the spatially varied nature of $H(f_x, f_y)$, where in each pixel can have its own unique PSF. Without an accurate model of the distortion present, the original clean image cannot be fully recovered. This is therefore an ill-posed problem, whereby, for a given distorted image I_{out}, there exist an extremely large number of possible clean images I_{in}.

A widely used technique for this problem is the use of video sequences, as opposed to single images. The added temporal information allows a better understanding of the clean latent image, and can greatly increase the performance of turbulence mitigation algorithms. Such methods can be loosely described by the diagram presented in the work from Anantrasirichai et al. [12], shown in Figure 2.11. In this, different algorithms/frameworks may follow different paths as shown by the different coloured arrows, applying different functions for each block. Such frameworks most commonly represent a many-to-one scenario, whereby the extra information provided by a video sequence is utilised to recreate a single high quality output [51].

'Image Selection' refers to the analysis of each frame, where the frames that are of the best quality are identified and passed on to the next stage of the framework. The motivation for this block is that it is assumed that, for each edge/feature in the scene, at least one frame in the sequence will present a clean representation. Such techniques were introduced by Gregory et al. [89] and is

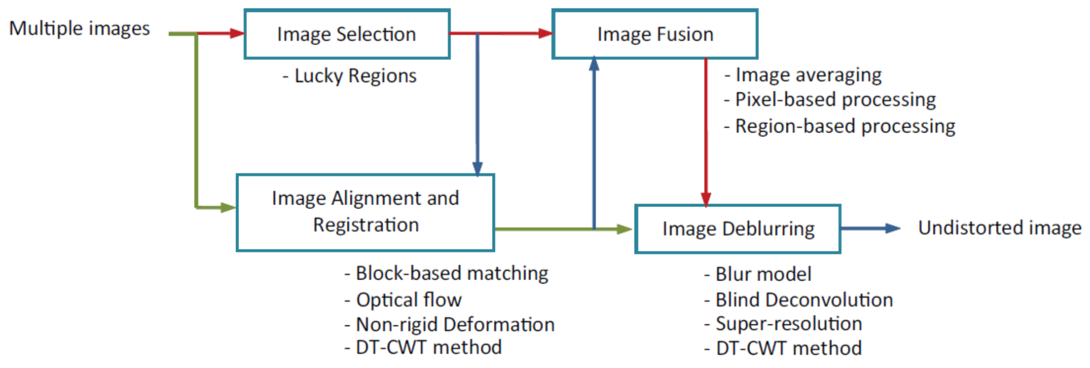


Figure 2.11: Flow of operations for traditional turbulence removal techniques [12]

known as 'Lucky Imaging'. The 'high quality' frames can be selected with the use of an image quality metric, as in the work by Huebner et al. [90], who measure both edge based and variance based metrics to identify high quality frames. Joshi et al. [91] make use of three separate factors to obtain a 'per pixel' quality weighting, these factors being sharpness, resampling error and the presence of dust. In the work from Wen et al. [92], they compute the average bicoherence of image regions, where the regions with the lowest average are deemed as high quality. Alongside traditional sharpness metrics, the work from Mao et al. [93] introduce an additional metric for geometric consistency, which ensures that any artifacts in the images do not report false sharpness patches. Another interesting approach to image selection is found in the work by He et al [94], who make use of Robust Principal Component Analysis (RPCA) to separate frames into a high-part and sparse part, where the former represents the general structure of the scene, and the latter contains the higher frequency texture information. By applying adaptive thresholding, high quality patches are found from each input frame [95].

Another method of processing the input sequence is with the 'Image Alignment and Registration' block. Instead of identifying 'high quality' frames, this block aims to align each frame to compensate for spatial inconsistencies over time. In this, the output is a sequence of frames that have been aligned temporally, therefore having any spatial distortions minimised. One technique that can be employed for this task is that of block matching [46, 96] where, in the work by Huebner et al [96] for example, it is used to align patches within each frame with a reference, before performing temporal averaging. This technique was further improved upon in the work by Rucci et al. [74], who developed an iterative block matching algorithm that generates a new reference image each iteration, using the registration information from the previous iteration. In the works by Zhu et al. [70, 97], a B-spline registration algorithm is introduced and utilised, in which a reference image is obtained by averaging the sequence of input frames. The geometric distortion between each individual frame and this reference image can then be calculated using B-spline interpolation.

The job of the 'Image Fusion' block is then to take a sequence of frames (either 'high quality' frames, or aligned frames) and fuse them into a single latent image. By making use of the added temporal information, this block is able to output a single image that represents the input scene with no spatial warping, with only the average distortion remaining. The most simple method of such fusion is the temporal averaging of the input frames [90–92], which would result in an image where the average PSF would represent a spatially invariant blur. An example of a pixel-based fusion method can be found in the works by Hardie et al. [15, 17], who first perform frame-wise alignment using interpolation. During this process, the distances between original and interpolated pixels are retained. The fusion step then makes use of these displacements as weights, where pixels with a larger displacement are weighted less than pixels with small displacements. In this, the pixels that did not require significant altering are reserved, as they were more likely to be originally correct. Another method of image fusion can be found in the works by Anantrasirichai et al. [13, 57], who utilise the shift invariance, orientation selectivity and multiscale properties of the Dual Tree Complex Wavelet Transform (DT-CWT) to perform region based image fusion at the feature level (as opposed to pixel level).

Finally, the 'Image Deblurring' block is responsible for removing any remaining distortions present in either the latent image (from the 'Image Fusion' block) or the aligned frames (from the 'Image Alignment and Registration' block). At this stage of the framework, it can be assumed that any warp induced by turbulence has been corrected, and the only remaining distortion is that of blurring. The output is therefore a clean image or a sequence of clean images. A common approach to such deblurring is the application of a Weiner filter [46, 98, 99], which makes use of the atmospehric OTF (Eq. (2.58)) to restore a clean latent image. Similarly Chimmit et al. [16] and Mao et al. [93] make use of the well defined PSF of turbulence to perform blind deconvolution.

2.6 Conclusion

This chapter has provided an in depth analysis of turbulence for the application of long range imagery. Presented first was a comprehensive introduction to turbulence theory. Due its complex nature, various assumptions can be made in regard to the statistical characteristics of a turbulent atmosphere, leading to definitions such as the modified von Kármán PSD. The key characteristics of turbulence were also defined, where the Fried parameter, isoplanatic angle and log-amplitude variance provide useful insights into the nature of a turbulent volume over a defined propagation distance. The Fresnel diffraction equation was also presented, facilitating a multi-step propagation method that is able to simulate the propagation of a point source through free space. The effects of a camera system were also defined, where it was shown that the aperture of a camera imposes a diffraction limit to the captured image. Such an aperture projects an Airy disk onto the camera sensor which, if larger than the pixel sensor, will result in blurring. The transfer function of such a lens was also described, where it was derived that a complex plane at the lens can be transferred to the focal length of the system with a simple Fourier transform.

As well as the background theory, this chapter also presented an analysis of the current literature in turbulence imaging. Firstly, existing datasets that consist of real life turbulence were summarised. Such datasets are used for tasks such as simulation validation, mitigation testing or turbulence characterisation. The various different methods used to simulate the effects of turbulence were then presented, as well as the datasets they have produced. Such simulations allow any level of turbulence to be applied to clean images, and are able to do so at a large scale. They also allow a ground truth image to exist for each distorted one, a feature that is of great use for mitigation algorithms that require the distortion free counterpart, such as deep learning. Finally, a summary of traditional (non deep learning) turbulence mitigation approaches were provided. It was shown that the majority of these algorithms follow a similar framework, by making use of multiple input frames and performing a combination of image selection, alignment/registration, fusion and deblurring to generate a clean latent output image. With the information presented in this chapter, the necessary background on turbulence for the remainder of the thesis has been established. Especially in regard to the development of the turbulence simulator described in Chapter 4.

Chapter 3

Deep Learning and Image Quality Metric Analysis

3.1 Introduction

The ability to automate tasks such as classification and denoising is extremely desirable as computer algorithms are able to perform extremely fast calculations, therefore facilitating much more efficient and streamlined processing frameworks. For example, the ability to denoise an image allows for cleaner images to be passed on to downstream tasks such as classification, ultimately improving the performance of full computer vision pipelines. In recent years, these processes have been dominated by machine learning architectures, as they have consistently shown state-of-the-art performance and can be optimised for fast throughput.

Deep learning is a data-driven technique that aims to train an algorithm an underlying representation of the data being used. For example, to train a machine to identify the difference between a dog and a cat, it is simply shown hundreds of images of both until it learns to separate the two classes. This is achieved by penalising the model when it predicts poorly, and rewarding it when it predicts correctly.

Krizhevsky et al. [19] demonstrated the power of Convolutional Neural Networks (CNNs) for image classification with AlexNet in 2012, achieving a dramatic improvement over traditional computer vision algorithms. This work is widely recognized as the beginning of the deep learning revolution in computer vision. Subsequent research has leveraged CNNs and other Deep Learning (DL) architectures to tackle a variety of tasks—including classification [21], segmentation [22], deblurring [23], and super-resolution [7]. The success of these models is closely tied to advances in parallel computing hardware, particularly GPUs, which have enabled the training of increasingly deep and complex networks.

Another application that has seen an improvement from DL is that of turbulence mitigation. Whilst only sparsely reported at the beginning of this research, its application has become the standard for state-of-the-art turbulence mitigation. This field has also taken inspiration from its traditional roots, and commonly makes use of turbulent video sequences, as opposed to single images. This allows the networks to gain a better understanding of the 'image behind the turbulence' (or latent image), and generate a turbulent free image accordingly. As detailed in Chapter 2, the largest issue facing the field of turbulence mitigation is the difficulty of data acquisition. A problem that is further amplified when applying machine learning techniques.

Another issue that faces the turbulence mitigation community is that of consistency between literature, including common datasets and common metrics. Whilst work continues in the development of simulation techniques and real life data acquisition, the use of metrics throughout literature is inconsistent. The most commonly used metrics are that of Peak Signal to Noise Ration (PSNR) and the Structural SIMilrity (SSIM) index. Both of which have been adopted from traditional image processing tasks. However, the task of turbulence mitigation, as with many other image processing tasks, is the streamlining of downstream task such as classification. It is therefore important that the metrics used to evaluate the quality of deep learning models reflect this ultimate goal.

The choice of such a metric is a difficult task, as image quality assessment (IQA) itself is a very popular and well researched field. There are therefore a great number of potential metrics to choose from, including traditional statistical based methods, as well as more modern DL based methods. Some works aim to ease this burden of choice by developing IQA datasets for the specific purpose of metric comparison; identifying which best agrees with human perception. In this, many metrics have been highlighted throughout the years as state-of-the-art, and have become more common in similar literature.

The remainder of the chapter is organised as follows. Section 3.2 first presents the background knowledge on machine learning and its evolution into the deep learning used today. It provides details regarding the design of DL architectures, as well as their training process. Section 3.3 then provides more detail into the use of DL for the specific application of machine vision, highlighting key developments in architecture design, and popular DL models. Section 3.4 provides an overview of the current literature in turbulence mitigation with deep learning, detailing the different approaches that have been employed. Section 3.5 provides a background of statistical analysis that is used throughout the thesis, especially for metric analysis, which is further presented in Section 3.6 which provides an overview of metric investigations in literature, as well as a high level description of the metrics used throughout the thesis. Section 3.7 then concludes the chapter.

3.2 Deep Learning

3.2.1 Conventional Neural Networks

The human brain is exceptionally good at a great many tasks, especially in the domain of processing visual signals. It is not difficult for a child to identify a dog, or for a adult to track and catch a ball. These tasks are performed instinctively, without thinking. It is however very difficult for a computer to perform similar tasks. According to a computer, an image of a dog is simple 0s and 1s, and it does not have the capability to understand what a 'dog' is, let alone if it is present in a picture. To allow a computer to gain an understanding of images and videos, or at least gain a mathematical representation of such, algorithms can be 'trained' to perform various tasks. This is the basis of Artificial Intelligence (AI), where a complete AI system would be able to think for itself, and create new ideas. A more simple approach to AI is that of teaching a machine to perform a single task, based on repeated experiences, also known as Machine Learning (ML).

The task of ML firstly requires data with which to train an algorithm. For the task of classification for example, large datasets of images are necessary, containing a restricted number of classes. Within these datasets, not only are the images required, but also the correct label for each sample. The creation of such a dataset is therefore labour-intensive, and is a key problem in machine learning.

Once a dataset is available, it can then be passed into a machine learning algorithm in batches, where initially, the algorithm will be clueless as to what it is seeing, and output gibberish. However, as the data is labelled, a penalty (or loss) function can be evaluated, which indicates how far away from the correct answer the algorithm is. Using this loss, the internal weights of the algorithm can be updated before being provided with the next batch of data. By performing these steps hundreds or thousands of times, the algorithm will slowly gain an understanding of its input, and be able to make 'educated' decisions regarding its output. All that remains the design of the algorithm itself.

The focus of this thesis is that of image processing, and therefore mainly makes use of Convolutional Neural Networks (CNNs). It is for this reason that only a brief description of more traditional Artificial Neural Networks (ANN)

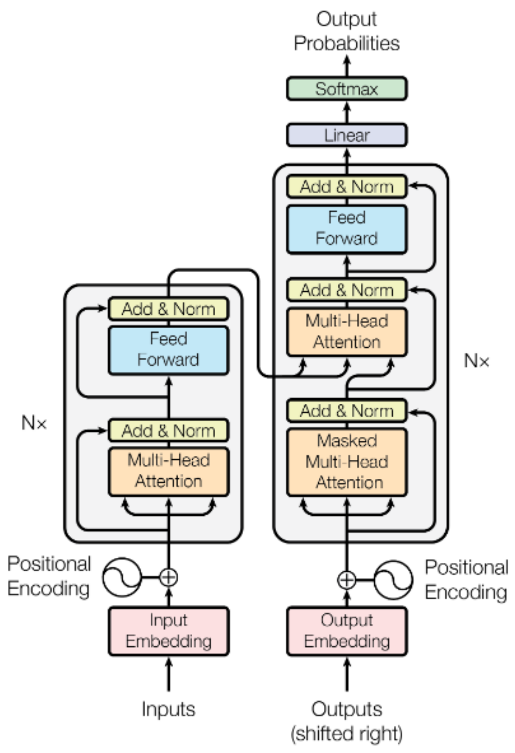


Figure 3.1: The Transformer architecture [103]

is provided here. Inspired by the architecture of the brain, Mcculloch et al. [100] introduced a model of a biological neuron which aimed to model neural activity with a binary threshold, whereby a neuron would ”fire” (output a 1) only if the weighted sum of its inputs exceeded a certain threshold, otherwise it would remain inactive (output a 0). In 1958, Rosenblatt et al. introduced the Perceptron, [101], which was then further improved upon by Rumelhart et al. [102], who introduced Multilayer Perceptrons (MLPs), non-linear activation functions and backpropagation. It is backpropagation that allows a model to ‘learn’ from its mistakes via error gradients.

3.2.2 Transformers

Sequence modelling is the task of predicting or generating elements in a sequence by identifying patterns and dependencies across its components. Such modelling can be performed on sequences of ordered data such as words or characters. This field has undergone significant advances in recent years, driven largely by the introduction of the Transformer architecture [103]. Compared to earlier recurrent approaches such as Recurrent Neural Networks (RNNs) [104] and Long Short-Term Memory (LSTM) models [105], Transformers offer improved performance and scalability across a wide range of sequence tasks.

The Transformer architecture consists of a stacked layers of encoders and decoders, as illustrated in Figure 3.1. Each of the encoder and decoder layers contains two main components: a self-attention mechanism and a feed forward network, with the decoder layers containing one additional component, the masked multi-head attention module. The task of the encoder is that of processing the raw input such that the decoder can use it to generate an output. It achieves this by allowing each element of the input to contextualise itself within the whole, therefore allowing the identification of key relationships and meaning. The task of the decoder is to take the contextualised output from the encoder and, in the case of the original transformer, perform text translation. To achieve this, the decoder generates output tokens one at a time, with each step being conditioned on both the encoder's output and the sequence of tokens it has already produced. This autoregressive process enables the decoder to produce contextually appropriate translations. The strength of the Transformer architecture lies in its ability to attend to the most relevant information within the input sequence. By identifying key relationships between words and phrases, it can generate an output that is both contextually coherent and semantically accurate, capturing long-range dependencies that traditional recurrent models often struggle to model. The introduction of the Transformer has led to a rise in large, pretrained models for language processing, commonly referred to as Large Language Models (LLMs). Such models are trained on vast amounts of text data from a great number of different sources including books, articles and websites. Each text sample is fed into the model, where it attempts to predict the next word in the sequence, and a loss function identifies how wrong the predicted word is. Over its training cycle, the model is slowly able to form full sentences based on its training data.

Whilst the work in this thesis does not directly implement text based sequence modelling, the processing of frame sequences is covered. The use of attention based mechanisms for such a task is explored further in Section 3.3.

3.2.3 Convolutional Neural Networks

The potential of CNNs was first shown in the work by LeCun et al. [106] in 2002, who used a CNN architecture to perform digit recognition. Then, as mentioned previously, the introduction of Alexnet [19] in 2012 constituted a major breakthrough in the use of CNNs for image processing tasks.

Unlike MLPs, which operate on 1D input vectors, convolutional layers apply learnable filters over 2D inputs, such as images, enabling spatial feature extraction. This is achieved with the use of 'convolutional kernels' which are typically

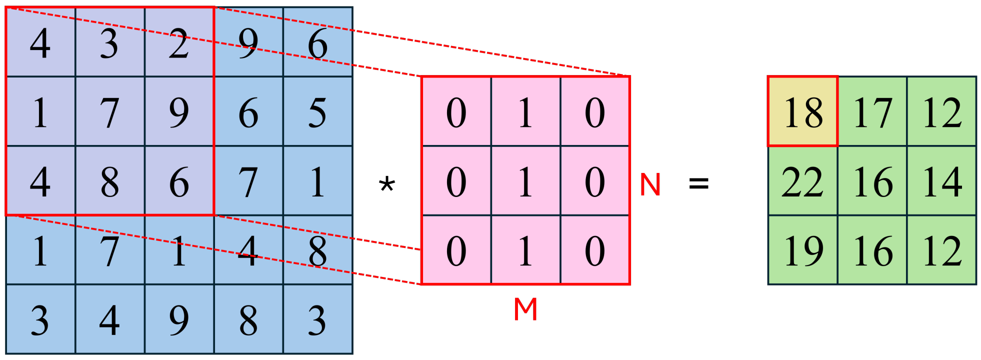


Figure 3.2: Illustration of a simple convolutional layer, with a 3x3 kernel and a stride of 1.f

of size 3x3 or 5x5. These kernels are first applied at the top-left corner of a given image, where an element-wise product is obtained before shifting the kernel across for the next calculation. This process is then repeated for all possible overlaps of the kernel and the input image. This process is illustrated in Figure 3.2, where the kernel is of size 3x3 and is applied to the input image with a stride of 1. The term ‘stride’ refers to the number of pixels the kernel is moved in between summing operations. As shown, due to the sizes of the input and kernel, a stride of 1 results in a reduced resolution of the output. To overcome this size reduction, it is common to first ‘pad’ the input, such that the output size matches that of the input. The mathematical representation of the convolution operation is given as,

$$Y(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K(m,n) \cdot X(i+m, j+n) \tag{3.1}$$

where, as seen in Figure 3.2, M and N are the dimensions of the convolutional kernel K and the term $X(i+m, j+n)$ represents the $M \times N$ region of the input image corresponding to the pixel at location (i,j).

The application of multiple convolutional kernels, allows each to inherently learn a specific feature during training. For example, one kernel may learn to look for edges, whilst another looks for textures. Given an input image with size $H \times W \times C$, where H, W and C are the image height, width and number of channels respectively, if a convolutional layer contains 64 kernels, the resulting output of that layer would be a feature map of size $H' \times W' \times 64$, where H' and W' are the new spatial dimensions, depending on the kernel size, padding and stride. Such layers can then be combined to form full CNNs, as shown by LeCun et al. when they presented LeNet-5 [106].

One issue with early CNNs however was that of backpropagation. Due to the combination of derivatives during this step, it was common that gradients would either vanish, or explode. This therefore limited the potential depth, and therefore representation power of models. A shift towards Deep Neural Networks (DNNs) was facilitated by the use of smaller kernels (3x3 instead of 11x11), which allowed more depth to be obtained. The final key breakthrough into DNNs came via skip connections, which allowed the gradient to reach all parts of the model.

DNNs have demonstrated versatility across numerous application domains. Consistently providing state-of-the-art models for machine vision that previously relied upon traditional mathematical algorithms. For example, denoising is a fundamental operation in image processing. The nature of image capture is extremely prone to additive noise, and commonly requires noise reduction. The utilisation of deep learning in this field is well established [107] and has indeed proven to outperform traditional techniques. Another application is that of super resolution, which aims to accurately resize images, whilst also increasing high frequency detail. The largest jump in performance was achieved by Dong et al. [108], who were the first authors to use CNNs for super resolution. In the years since, the use of deep learning has become the norm for super resolution, with an annual competition to assess super resolution networks on the same data [109, 110]. Due to the ever continuing growth of machine learning algorithms and the increased improvement and availability of computing power, more and more problems are being improved upon by the application of machine learning, especially in the case of image and video processing.

3.2.4 Basic Architecture Design

The range of techniques available for designing CNN architectures is extensive. Whilst specific methods have been proposed for specific applications (see Section 3.3), several architectural components remain fundamental across a wide variety of CNN models.

One such component is the activation layer. This is usually applied directly after a ANN or CNN layer, and introduces non-linearity into the model. Without such layers, the model would simply learn a linear function, and would not learn complex patterns. Common activation functions include the the Sigmoid function, tanh and the Rectified Linear Unit (RelU), all of which can be seen in Figure 3.3.

Another key design aspect of CNNs is the ability to downsample feature maps, a technique used since early models like LeNet-5 [106]. The two most popular methods of downsampling are max pooling and average pooling. In both cases,

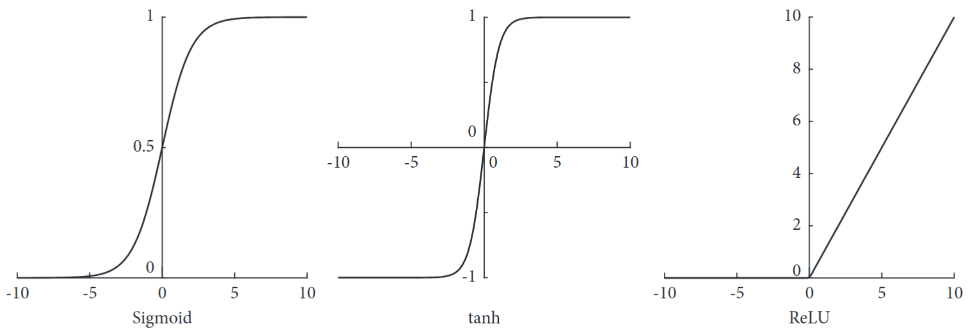


Figure 3.3: Different activation functions used to introduce non-linearity into models [111].

the pooling operation operates on small patches of the input feature map. Within each patch, either the max value or the average value is obtained, and used as output. A similar downsampling can be achieved via a larger stride during convolution.

A final technique used across ML literature is that of regularisation. Due to a models significant number of learnable parameters, it can quite easily overfit on a given dataset. Overfitting is a scenario where a model can perfectly process the training data, and can no longer be trained. Whilst this initially seems like a good thing, such a model has no ability to generalise to new data, as it has simply memorised the training data. To overcome overfitting, a number of techniques can be utilised, such as dropout. This method simply 'drops' a fraction of the weights each training iteration. This therefore discourages the network to rely on the same set of available weights, and encourages generalisation. Another regularisation technique is the augmentation of the input data, such as scaling, rotation and flipping. This essentially increases the size of the training dataset, making it more difficult for the model to memorise. A final regularisation technique is that of batch normalisation. This is applied to the output feature maps of linear or convolutional layers, where they are normalised such that they have a mean of zero and a standard deviation of one. This aids in training by regularising the inputs to layers i.e. layers learn that their input will have known statistics.

3.2.5 Model Training

The training of CNNs involves an iterative process in which the model is provided with the entire training dataset multiple times where each complete pass through the dataset is called an epoch. Within each epoch, the dataset is further segmented into chunks called batches. This is required, as the processing of an entire dataset would be computationally infeasible. Batch training also introduces another element of regularisation, as well as more stable gradient estimation.

The output of each batch is passed into a loss function, which determines the accuracy of the model at its current state. It is therefore vital that the loss function is designed such that the model is steered towards a correct solution. For tasks such as classification, a common loss function is that of Softmax [26], which is defined as,

$$\hat{y}_i = \operatorname{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3.2}$$

where $z = [z_1, z_2, ...]$ are the raw outputs from the classifier (also called logits), and the outputs represent the probability of each class. These probabilities are then passed to a cross-entropy loss, defined as

$$L = -\sum_i y_i \log(\hat{y}_i) \tag{3.3}$$

which penalises the model if the probability for the correct class is low.

For regression models, such as image processing tasks, the most common loss functions are those of the Mean Absolute Error (MAE) and Mean Squared Error (MSE), defined as,

$$L = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.4}$$

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.5}$$

where $\hat{y}_i$ is the model output and y_i is the corresponding ground truth. These are also referred to as the $L1$ and $L2$ distances respectively. Due to the square in MSE, any large errors are penalised more heavily. MAE on the otherhand is more robust to outliers. A final loss of interest is that of the Charbonnier loss [112]. This function aims to incorporate the robustness of MAE whilst also penalising large errors. It is defined as,

$$L = \sqrt{\|\hat{y} - y\|^2 + \varepsilon^2}. \tag{3.6}$$

When the error $\hat{y} - y$ is large, this loss behaves like the MSE, whilst when the error is small it behaves more like the MAE loss.

The task of a machine learning algorithm is to minimise the chosen loss function such that,

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f_\theta(x), y) \tag{3.7}$$

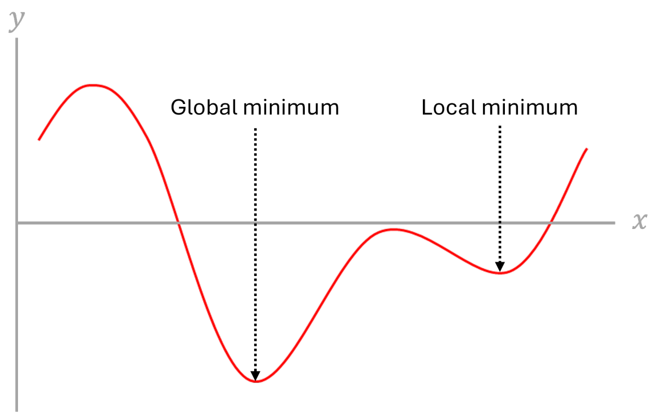


Figure 3.4: Simple two dimensional example of a local minimum, potentially restricting an optimisation from finding the global minimum.

where θ represents all the learnable parameters within the model, such as weights and biases. The term $f_\theta(x)$ is the output of the model given an input x and $\mathcal{L}\left(f_\theta(x), y\right)$ represents the resulting loss, given the model output and the ground truth y. The iterative process of batch training allows the weights θ to slowly 'walk' towards the ideal solution θ^* in a process known as Stochastic Gradient Decent (SGD).

To perform SGD, the gradient of each weight in the model is calculated as a function of the current loss, calculated as,

$$g_i = \frac{\partial \mathcal{L}(\theta)}{\partial \theta_i} \tag{3.8}$$

where g_i is the gradient associated with the parameter θ_i. Once the loss is evaluated, the resulting gradients throughout the model are then calculated. Each weight is then moved along the gradient direction by the defined learning rate, which is a tunable parameter during training. During this iterative process, the loss should continue to fluctuate (hopefully in a downward fashion) until it plateaus, where it could be considered that the minimum has been found, i.e. θ^* has been obtained. However, the 'loss landscape' formed by Eq. (3.7) is understandably complex and multi-dimensional, therefore the optimisation process could get 'stuck' in a local minimum where the gradient is zero. This can be problematic if the model's performance at this local minimum is worse than its potential optimal performance, or global minimum, as illustrated in Figure 3.4.

To reduce the chances of finding local minima, optimiser functions are equipped with several techniques to 'escape' such situations. One such technique is that of momentum, whereby the movement of the weights along the gradient retains a fraction of the previous update. This allows the optimiser to ‘roll by’ local minima or saddle points. The learning rate can also be adjusted to improve optimisation, as it is best to initially take larger steps towards the minimum, before reducing the step size in order to fine-tune the optimum value. One such optimiser that combines these ideas is that of ADAM [113], which keeps track of the momentum as well as the mean of squared gradients, where the latter captures the scale of the gradients such that the learning rate can be adapted to the ‘steepness’ of the gradients. These two moving averages are calculated as,

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{3.9}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{3.10}$$

where m_t, v_t and g_t are the first moment (Momentum), second moment (mean of squared gradients) and the current gradient at the time step t respectively. The parameters β_1 and β_2 control the level of decay present in each moving average, and are typically set to values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. To avoid the potential bias problems due to m_t and v_t being initialised at zero, they are first corrected as,

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.11}$$

where, due β_1 and β_2 being raised to the raised power of t, as training continues, the denominator slowly approaches 1, therefore reducing this bias correction. The parameters θ are then updated as,

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.12}$$

where η is the learning rate provided as a parameter to the model.

A final element of training a model is the selection of the various parameters that control the learning process, known as hyperparameters. These cover areas such as architecture design, training time and dataset control. The experimentation of these hyperparameters is a fundamental operation in deep learning research, as it is difficult to initially know what values of each parameter will result in an optimal model. For example, switching from 3x3 kernels to 5x5 kernels can increase the receptive field, potentially improving performance. Similarly, increasing the batch size can help the model form better representations and train more quickly. Potentially one of the most important hyperparameter is that of the learning rate η, as this is what controls the steps taken during gradient decent, and can easily restrict a model from training if chosen incorrectly.

Alongside this is that of the choice of learning rate scheduler, which controls how the learning rate changes during the training procedure. If the learning rate remains unchanged, the optimizer can only progress so far. Once a minimum is approached, a learning rate that is too large will cause the optimizer to overshoot and keep "jumping over" the optimal solution, a learning rate that is too small will simply take too long to reach the same optimal solution. A learning rate scheduler therefore dynamically alters the step size as training proceeds. Examples of different learning rate schedulers, and a brief description of each can be found in Table 3.1.

Table 3.1: Examples of learning rate schedulers.

Scheduler	**Description**
Step Decay	Learning rate drops a set amount at defined intervals (e.g. every 10 epochs).
Exponential Decay	Learning rate decreases exponentially over time, with a defined decay rate.
Cosine Annealing	Learning rate follows a cosine curve, gradually decreasing over time.
Reduce on Plateau	Learning rate is reduced when the loss has stopped improving for a number of epochs.
Warm Restarts	Learning rate is periodically reset to a higher value after decaying, often using cosine annealing between restarts.

3.3 Deep Learning for computer vision applications

3.3.1 Notable design techniques

The popularity of deep learning research inevitably lead to various breakthroughs in architecture design. Such key trends are detailed here.

The first key trend is the development, and use, of encoder-decoder architectures. These models provide the ability to map inputs to outputs by first encoding the input into a latent representation, which is then decoded to generate an output. This architecture differs significantly from classification models as, instead of reducing the input down to a single value, the encoder-decoder architecture aims to map an input to a desired output. For example, the task of image deblurring requires the output to be a clean, distortion free, version of the

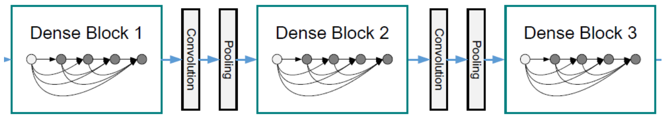


Figure 3.5: Example use case of dense blocks. Where within each block, each layer is densely connected, i.e. each layer is fed into all subsequent layers [116]

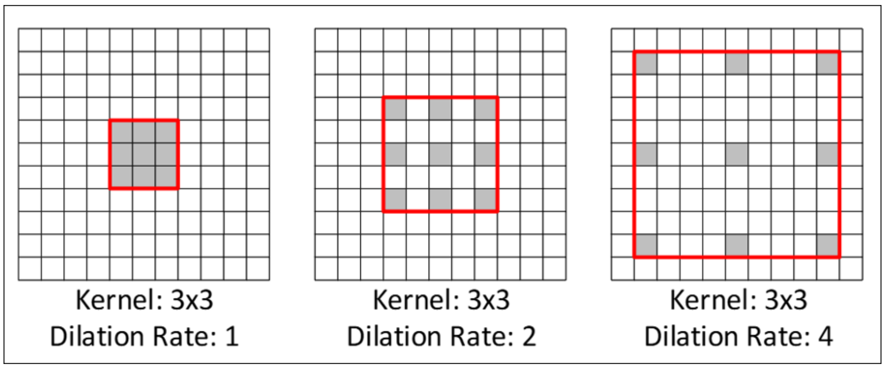
input. This can be achieved via the encoder-decoder architecture. One benefit of such a design allows a great deal of processing to occur within the latent space between encoding and decoding. Such processing allows the models to focus on the more important details of the input, and disregard redundant information. A sub-class of the encoder-decoder architecture is that of the autoencoder. This model simply takes an input x, and aims to produce an output $\hat{x}$ that is as close to the input x as possible. Such a model can then be used for dimensionality reduction by simply using only the encoder, where it can be assumed that the latent space represents a meaningful representation of the input. The ability of the encoder-decoder to transform inputs into structured latent representations before producing outputs has established it as a central approach in deep learning research, and can be found across many different fields.

The next trend in architecture design is that of residual learning [114], which is the inclusion of a skip connection that allows the input to the network to be combined with the output. Let the input to the network be a distorted image y:

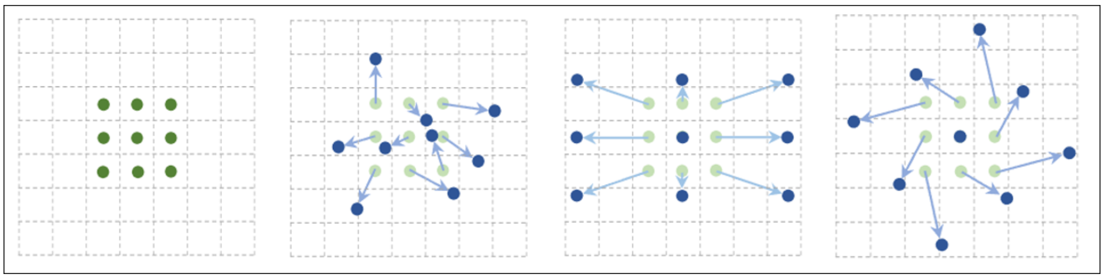
$$y = x + n \tag{3.13}$$

where x is the ideal, sharp image and n is the distortion applied to the image. The inclusion of a skip connection allows the network to simply learn the residual, n, which can then be subtracted from the original input x. This allows the network to gain a better understanding of the degradation process without any bias from the image contents. A further improvement on residual learning is the use of residual 'blocks' [115], where residual learning is performed over smaller segments within the model. This allows the network to be far deeper, as the connections allow a 'highway' for the backpropagation of the loss.

Inspired by the success of residual learning, another common practice in deep learning architectures is the use of densely connected layers, first introduced by Huang et al. [116]. This type of architecture sees the output of each layer being fed as an input to each subsequent layer, as illustrated in Figure 3.5. As with residual blocks, this aids backpropagation and also allows each layer to gain a better understanding of the layers preceding it.



(a) Illustration of two dilated convolution operations, with varied dilation rates. Such convolution allows a larger receptive field [117].



(b) Example cases of deformable convolutions, where the offset vectors adapt to the image contents, and allows the kernel to access pixels outside the traditional 3x3 grid [118].

Figure 3.6: Two key alternatives to traditional 3x3 kernel convolutions: Dilated convolution and Deformable convolution.

As well as changes made to how each layer is connected to one another, different forms of convolution have also been presented. Whilst the most typical convolutional kernels are 3x3 or 5x5, it is also beneficial to use simple 1x1 convolutions. This allows the number of features in a model to be combined without simply discarding information. More complex variations include dilated convolutions [117] and deformable convolutions [118, 119], as shown in Figure 3.6. In the case of dilated convolutions, gaps are introduced between the traditional 3x3 kernel grid, allowing a single convolution to access a larger area of the input image, therefore increasing the resulting receptive field. Deformable convolutions on the other hand, permit each each kernel weight to move around freely. Such movement is governed by learnable offset and modulation weights, where the modulation mechanism was introduced by Zhu et al. [119]. The deformable convolution allows a model to dynamically adapt to image contents, and place more attention on the areas of interest in an image. As shown in Figure 3.6b, this also allows the layer to correct for scaling and rotations.

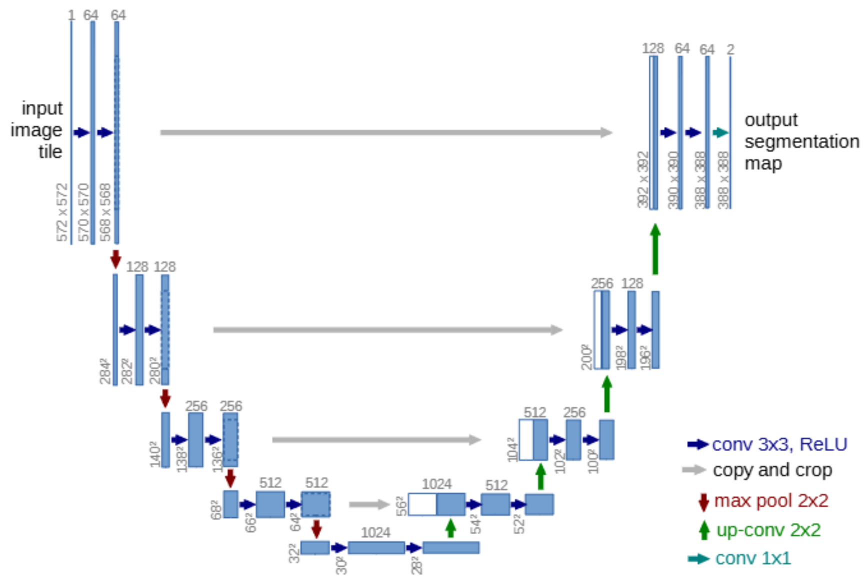


Figure 3.7: U-net architecture that employs an encoder and decoder structure. Skip connections link the levels of the encoder to the corresponding levels of the decoder [121].

3.3.2 Notable architectures

Throughout this thesis, several deep learning models are mentioned and utilised. It is therefore worth providing a brief overview of each, and highlighting their importance in the field of machine learning.

Firstly, several classification models have been proposed in literature as, since the introduction of Alexnet [19], convolutional models have provided state-of-the-art performance. Such models include VGG19 [120], ResNet-50 [114] and EfficientNet [21], each representing a step forward in model design as deep learning architectures evolved with continued research. The simplest architecture, VGG19, makes use of 19 convolutional layers, applied to periodically downsampled features, followed by 3 fully connected layers to predict classifications. The slightly more complex architecture of ResNet-50 makes use of 49 convolutional layers followed by a single fully connected layer. They key change that ResNet introduced was the use of skip connections, facilitating improved backpropagation. Finally, EfficientNet improves on earlier models by carefully balancing how deep, wide, and high-resolution the network is, and uses efficient building blocks and attention mechanisms to focus on the most important features in an image. It therefore represents an extremely accurate and efficient classification model

While classification models can accurately identify the contents of a scene, there are times when the goal is not just to classify an image but to modify its contents. As covered in Section 3.3.1, such a task can be achieved via encoder-decoder architectures. One such architecture that is widely adopted is that of U-net from Ronneberger et al. [121], which was originally designed for the segmentation of neuronal structures in electron microscopic stacks. It is fully convolutional and contains connections between each scale of the encoder and the corresponding scale of the decoder, as seen in Figure 3.7. Such skip connections prevents the loss of spatial information during downsampling.

Another desirable domain for deep learning is that of image generation. The ability to generate a brand new, unseen, images allows for a great deal of creative freedom, especially if able to direct the output image using a text prompt. The original synthesis of such images, without a text prompt, was revolutionised with the introduction of Generative Adversarial Networks (GANs) by Goodfellow et al. [122]. In this work, Goodfellow proposed the use of two networks: A generator and a discriminator. These two networks would be trained in parallel, and each would act as the others opposite. The purpose of the generator was to take a noise vector as input, and generate a realistic looking image. The job of the discriminator on the other hand was to assess these generated images and classify it as fake. By supplying the discriminator with real life images, as well as the generated ones, it would be able to learn to identify the difference. However, as the discriminator learns, the generator also learns to generate better looking images. The two models therefore continue to improve throughout training until, ideally, the generator can synthesise perfect images, and the discriminator can perfectly distinguish between real and fake, this is known as a Nash equilibrium. The original GAN was further improved upon by the introduction of the conditional GAN from Mirza et al. [123]. This new model allowed an additional input, alongside the random noise vector. This new input was a condition, or label, that the GAN was to work towards. For example, if generating digits, this label could instruct the model to generate a specific digit. An additional improvement to the original GAN was introduced by Arjovsky et al. [124], who introduced the Wasserstein GAN, which stabilised the training process and reduced the probability of mode collapse. Where mode collapse is a state where the generator learns to only generate a single image, one that it knows will fool the discriminator, therefore simply ignoring the random input vector.

In recent years, the application of deep learning to computer vision tasks has advanced significantly, largely due to the introduction of the Transformer architecture [103].For example, in 2020, Dosovitskiy et al. introduced the Vision Transformer (ViT), adapting the Transformer architecture for image recognition [125]. This was achieved by first partitioning an input image into patches of

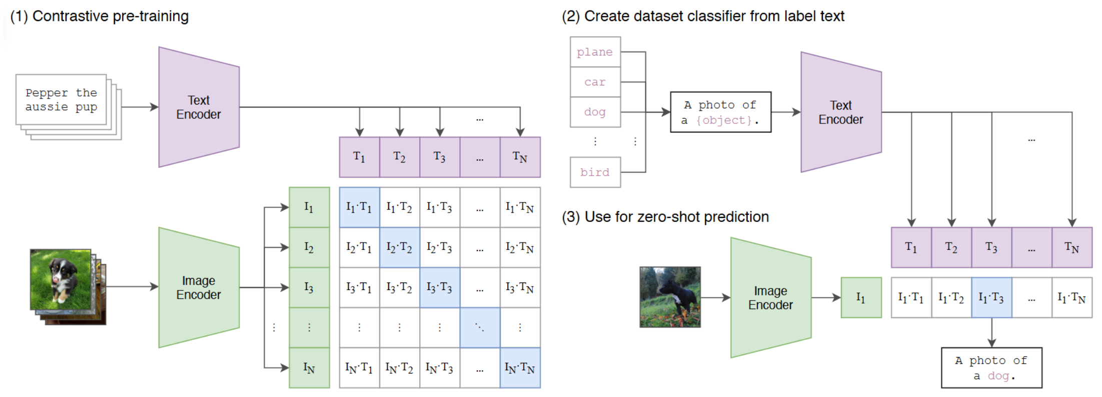


Figure 3.8: Overview of CLIP architecture. First the image and text encoders are trained with correct image/text pairings. During test time, the optimal prompt is identified for the given image [127]

.

size 16x16, which are then flattened into a 1D vector. These vectors are then embedded with patch and position encodings before being passed, as one, into a standard Transformer encoder. The output of which can then be processed by an MLP layer to provide a classification prediction. This process allowed the ViT to attend to different spatial locations within the image, allowing a more comprehensive understanding of the image contents. By treating the image as a sequence of patches rather than using convolutional operations, the ViT can capture long-range dependencies and global context more effectively than traditional CNNs, enabling improved performance. Significant advancements of the ViT architecture include the Swin Transformer [126] and Restormer [23], notable due to their use for image processing tasks such as segmentation and deblurring. The Swin Transformer introduces a hierarchical design with Shifted Window-based Multi-head Self-Attention (SW-MSA) blocks, which allow for greater efficiency by limiting the attention mechanism to non-overlapping local patches. The Restormer architecture makes use of two core modules: multi-Dconv head transposed attention (MDTA) and a Gated-Dconv feed-forward network (GDFN). Where the former performs attention over channels, as opposed to the spatial domain, and utilises depth wise convolutions (Dconv) to emphasise local context. The GDFN also makes use of Dconv layers to perform feature transformation

As well as simple one-off classification, it is also highly desirable to gain a natural language description of what is in an image, especially for automation purposes. The ability to provide specific details such as "A photo of a golden retriever playing in the grass" allows more flexibility away from predetermined classes. The most common model for this task is that of CLIP (Contrastive

Language–Image Pretraining) from Agnolucci et al. [127], which takes a series of prompts and an image as input. Using an image encoder (Such as ViT) and a text encoder (Such as a Transformer), embeddings are created for each, which are then compared using cosine similarity. This process identifies which text embedding best matches with that of the input image, which is then presented as the output. An overview of the CLIP architecture is illustrated in Figure 3.8.

Another innovation in recent years is the introduction of diffusion models. Although GANs dominated state-of-the-art image generation for some time, continued advances in deep learning inevitably led to the emergence of new techniques. One such method is the Denoising Diffusion Probabalistic Model (DDPM), introduced by by Ho et al. in 2020 [128], which is able to greatly exceed GANs in image generation quality. The principle behind diffusion is to gradually add noise to training images over many steps, and train a model to reverse this process by denoising at each step. During training, the model learns to predict either the original clean image or the noise added at each step. Once trained, the model can start from pure noise and iteratively denoise it to generate a realistic image. Such a technique was then be further evolved to perform text-to-image diffusion, such as DALL-E [129]. In this, a text prompt is encoded using a technique like CLIP. Then as diffusion is taking place, the model can attend to this encoding, therefore steering the final output towards the desired prompt.

3.4 Deep Learning for Turbulence Mitigation

Whilst the majority of turbulence literature attempts to mitigate the distortions present, some approaches simply apply deep learning for tasks without such mitigation. It is argued that, by retraining models *with* turbulent data, the use of two separate techniques (mitigation then classification) can be replaced by a single model (classifcation *through* turbulence) [130]. For example, Hu et al. [131] make use of a multi-scale feature extractor and region proposal network to classify object types in turbulent environments. LeMaster et al. [132] make use of pooling layers to reduce an input image into features, which are then passed into more traditional ANN layers for classification. The recent work from Piscaer et al. [24] uses the EasyOCR engine to transfer a text recognition model to the field of turbulence, and shows promising results.

Another use of deep learning for turbulence image data is that of turbulence estimation, whereby the task of the model is to identify the level of turbulence present, as opposed to its mitigation. One such model is proposed by Saha et al. [62] who propose two different approaches. The first approach is fully convolutional, and simply takes a sequence of frames, and outputs a predicted C_n^2 value. The second approach combines deep learning and known imaging priors such as the camera settings, where such knowledge is utilised alongside a CNN architecture to estimate C_n^2. Similarly, the work from Celik et al. [133] make use of a neural network to estimate C_n^2 using only meteorological data such as temperature and humidity.

Before detailing the trends in 'end to end' approaches, whereby models perform the full mitigation process, it is worth highlighting deep learning networks that have been integrated into traditional frameworks by replacing the 'image deblurring' block shown in Figure 2.11. An example of such an approach can be found in the work by Hoffmire [134], who modify the BM-WF method from Hardie et al. [46] by replacing the Wiener Filter (WF) with a convolutional neural network, therefore presenting BM-CNN. Nieuwenhuizen et al. [56] perform a similar replacement of the blur estimation and unsharp masking steps in [135] with a DnCNN model [136]. As well as deblurring, the other steps of traditional frameworks can also be improved with deep learning, as in the work from Fazali et al. [137], who employ an encoder to extract features for frame selection as well as performing fusion with an autoencoder architecture.

The application of deep learning for turbulence mitigation, from a data perspective, makes use of both single frame [4, 60] and multi-frame sequences [59, 138, 139] where, in line with video processing techniques of the time, video frames are first stacked prior to being processed. This was a shift away from RNN type architectures that were used before [140].

One of the earliest uses of deep learning for turbulence mitigation was that from Chak et al. [25], who were the first to identify the potential improvements over traditional approaches that DL could provide. Using the two methods, SGL [141] and IRIS [14], Chak et al. showed that their Turbulence Removal Network (TRN) was able to provide significant improvements. This finding was then confirmed in future works, such as Gao et al. [66] and Jin et al. [59], both of which present a deep learning model that is able to outperform the chosen traditional algorithms. This pattern has continued throughout research [87, 142–144], where modern papers no longer compare against traditional metrics, and aim to simply improve upon the results reported from other DL based models [60, 88, 145–149].

With regard to architecture design, it is common for pre-existing architectures to be repurposed for the task of turbulence mitigation. The U-net [121] architecture for example is utilised in a great number of turbulence mitigation approaches [4, 36, 37, 40, 76, 148, 150], where the architecture is commonly used as a backbone for more complex techniques. The work from Xia et al. [148] for example, make use of a U-net to extract frame-wise feature maps for the input sequence, before being passed onto their Output-By-Group module for feature fusion. Another example can be found in the work from Li et al. [150], who stack two U-nets together, with a supervised attention block placed in between, to form their DeblurNet.

Uses of more traditional CNN architectures have also been adopted, and further adapted to better suit turbulence mitigation [33, 66, 143, 151, 152]. The work from Guo et al. [152] for example, make use of traditional convolutional layers and dilated convolutional layers for feature extraction, as well as densely connected layers for feature fusion. Another interesting use of CNN layers can be found in the work by Whyte et al. [153], who propose two networks: a grid deformer and an image generator, both of which are composed of convolutional layers. In this, the grid deformer takes the input frames and estimates the deformation due to turbulence, which it then applies to a uniform grid. This grid is then fed to the image generator, which aims to undo the resulting distortion field.

The work by Anantrasirichai et al. [139] expand on traditional CNN architectures by employing complex-valued convolutions where it is proposed that, due to the phase shift induced by turbulence, such layers are better able to remove atmospheric distortions. Similarly, two works by Khowaja et al. [154, 155] make use of the Discrete Wavelet Transform (DWT) in their architecture to perform image registration. An approach that resulted in 2nd place [154] in the 2022 UG^2+ challenge for turbulence mitigation.

Another popular approach to turbulence mitigation is the utilisation of GANs [25, 59, 75, 144, 145, 156], especially in the field of turbulence mitigation for facial recognition [38, 39, 138, 157], where their ability to generate photo-realistic images is utilised. The Atmospheric Turbulence Face GAN (ATFaceGan) from Lau et al. [38] makes use of two separate generator model: one to correct distortion and one to perform deblurring. These are applied in two different permutations along two paths, before being fused together by a final generator model. To train these three models in parallel, three seperate discriminators are also utilised, one for each generator. This methodology allows the authors to ‘disentangle’ the two distortions of blur and turbulence distortion. An example usage of GANs for natural scene data can be found in the recent work by Lu et al. [156], who develop a generator network with multi-scale feature extraction, and employ a VGG architecture for their discriminator network [120].

More recently, image generation techniques have shifted away from GANs, and towards diffusion models, which are far more capable at generating realistic output images. The DDPM architecture has also been adopted by the turbulence mitigation community [158, 159], with Nair et al. [11] being the first to apply diffusion for facial recognition and Wang et al. [158] applying diffusion to natural scenes.

As well as diffusion, transformer architectures have also become widely used in machine vision tasks [35, 60, 88, 160, 161], where the work from Sun et al. [159] actually combines these two techniques for their Probabilistic Prior Turbulence Removal Network (PPTRN). Whilst diffusion models can provide state-of-the-art generation techniques, transformers are better able to capture the context of the input, and understand the spatial and temporal correlations. A common transformer architecture repurposed for turbulence mitigation is that of Restormer [23, 87, 162, 163], where its attention mechanism allows the high quality regions of the input sequence to be prioritised. In fact, even when not using Restormer directly, its influence can still be seen in other works [87, 160]. The use of attention mechanisms, which is the core principle of transformer networks, is an extremely useful tool capable of incorporating useful information accross channels. The DATUM architecture from Zhang et al. [88] make use of attention in their Multi-head Temporal-Channel Self-Attention (MTCSA) module to perform feature fusion. The recent work by Yin et al. [35] also made use of attention in their physics-based Turbulence Mitigation network guided by Turbulence Signature (TMTS) approach for the improved observation of the earth from space.

Recently, several works have made use of MAMBA from Gu et al. [164] for turbulence mitigation, such as Hill et al. [149], who present a 3D Mamba-Based Atmospheric Turbulence Removal (MAMAT) model. Similarly, Zhang et al. [147] present their MambaTM model. In both works, the results presented outperform recent state-of-the-art approaches such as DATUM.

3.5 Quality Analysis

When assessing the quality of an algorithm, the idea of what constitutes as ‘good’ is often highly subjective, especially in the case of image processing. It is therefore common to obtain human subjective scores for datasets. Such a collation of judgements is known as a Mean Opinion Score (MOS) [27], which can be obtained in many different manners, such as asking users to choose between two images [165] or by assigning each image with a score between 1 and 5 [166].

Once the MOS has been obtained, it is common to identify metrics that best agree with human perception. To assess the agreement between each metric and the MOS, several correlation coefficients have been developed, namely the Pearson Linear Correlation Coefficient (PLCC) [167], Spearman's Rank Correlation Coefficient (SRCC/SROCC) [168], and Kendall's Rank Correlation Coefficient (KRCC/KROCC) [169], where each coefficient emphasises different aspects of the relationship between the metrics and the MOS.

PLCC is calculated as

$$\text{PLCC} = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} \cdot \sqrt{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}} \tag{3.14}$$

where n is the number of observations, x_i and y_i are the values of the two lists (metric scores vs MOS), $\bar{x}$ and $\bar{y}$ are the means of x and y. This coefficient is able to measure the strength and direction of the linear relationship between metric scores and MOS. The PLCC has a value of -1 for a perfect negative correlation, and $+1$ for a perfect positive correlation. SRCC is calculated as

$$\text{SRCC} = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n\left(n^2 - 1\right)} \tag{3.15}$$

where n is the number of observations, $d_i = R\left(x_i\right) - R\left(y_i\right)$ and $R\left(x_i\right)$ and $R\left(y_i\right)$ are the ranks of x_i and y_i. This coefficient measures the strength and direction of the monotonic relationship between metric scores and MOS. Unlike the PLCC, the SRCC first ranks the two sets of scores ($R\left(x_i\right)$ and $R\left(y_i\right)$), and then finds correlation. KRCC is calculated as

$$\text{KRCC} = \frac{C - D}{\sqrt{\left(C + D + T_x\right)\left(C + D + T_y\right)}} \tag{3.16}$$

where C is the number of concordant pairs, D is the number of discordant pairs and T_x and T_y are the number of tied pairs in x and y, respectively. Concordant and discordant pairs are determined by comparing the orderings of two variables, where a pair is concordant if the order is the same in both variables, and discordant if the order is reversed. This coefficient also makes use of ranked scores and is more robust to ties.

Whilst correlation scores such as SRCC and KRCC can provide a measure of agreement between metric scores and MOS, it can also be beneficial to examine

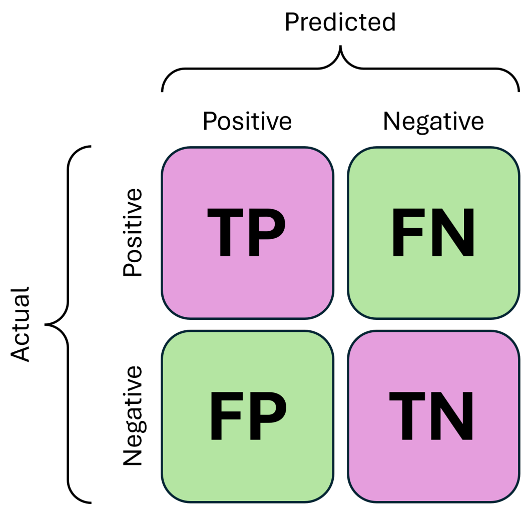


Figure 3.9: Illustrating of a confusion matrix, showing True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).

how these metric scores relate to a binary classification task. If, for example, a metric were able to reliably assign higher scores to correctly classified images and vice versa, it could be used as a classification 'proxy', indicating the likelihood of correct classification. An issue with this approach however is that, when training a classification algorithm, it is important to provide it with an equal distribution of all classes. However, sometimes it is inevitable that one class is more common than another, resulting in a classification bias. Take the task of animal classification for example. If 10% of the images are dogs whilst 90% are cats, there is a clear imbalance of representation. To reduce the bias caused by this imbalance, the evaluation metric of Balanced Accuracy (BAcc) can be utilised. This metric aims to balance the True Positive Rate (TPR) and True Negative Rate (TNR), and represents a probability of classification accuracy. The TPR and TNR are derived from the four components of a confusion matrix: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), as shown in Figure 3.9. The TPR and TNR are calculated as a function of a threshold τ as:

$$\text{TPR}(\tau) = \frac{TP(\tau)}{TP(\tau) + FN(\tau)}, \quad \text{TNR}(\tau) = \frac{TN(\tau)}{TN(\tau) + FP(\tau)}.$$

The Balanced Accuracy is then defined as [170]:

$$\text{Balanced Accuracy}(\tau) = \frac{\text{TPR}(\tau) + \text{TNR}(\tau)}{2} \times 100. \tag{3.17}$$

Where the value of τ can be altered until an optimal threshold τ^* is found such that

$$\tau^* = \arg\max_{\tau} \text{Balanced Accuracy}(\tau), \tag{3.18}$$

3.6 Metrics for Image Quality Assessment

In tasks such as tracking, the error between the predicted location and the ground truth is an easy calculation, it is therefore trivial to identify if an algorithm is performing well. For image processing tasks, such an assessment is far more difficult. The quality of an image is extremely subjective, and can rely on multiple factors such as sharpness, contrast, composition or colour grading. Whilst most human observers would be able to distinguish between blurry and sharp images, it is difficult to obtain such an observation via computer algorithms. In domains such as image compression for transmission, it is vital that the reconstructed images matches the original as accurately as possible. Therefore, when designing image processing algorithms, metrics of image quality are also required. Such image quality assessment (IQA) metrics mainly come in two forms: Full Reference (FR) and No Reference (NR). Where in the case of FR metrics, the original, ground truth, image is available alongside the processed image, therefore allowing a direct comparison to be made between the two. In the case of NR metrics, the ground truth image is unavailable, so the metric aims to measure image quality based on information extracted from the processed image alone.

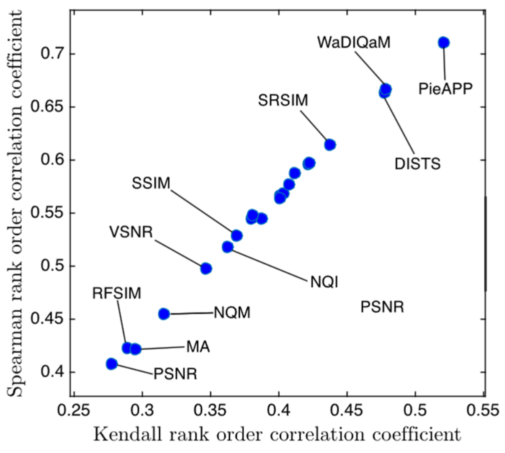
The most commonly used metrics in literature are that of the FR metrics Peak Signal-to-Noise Ratio (PSNR) and Structural SIMiliarity (SSIM) [27, 165, 171], as they are both informative and efficient to compute. However, it is argued that, in some image processing applications, the pervasive use of these metrics is ill-suited. The work from Blau et al. [172] perform an investigation into the trade-off between ‘distortion’ and ‘perceptual’ quality. Where they define ’distortion’ quality as the dissimilarity between an image and its ground truth, and ‘perceptual’ quality as the visual quality of an image, regardless of its similarity to the ground truth. In this work, Blau et al. comment on the misuse of distortion measures such as PSNR and SSIM, as they are potentially overlooking visual quality [173, 174], an aspect that some applications would prefer over reconstruction quality [115]. For example, it is commonly agreed that metrics such as PSNR and SSIM are at odds with perceptual quality [115, 173, 175–177]. Whilst it is true that an exact match with the ground truth would imply a ’good’ image [178], such a reconstruction is only one possible solution, and better solutions may exist that would be penalised by FR metrics [176]. It is therefore recommend by Blau et al. that, when assessing the quality of images, multiple IQA metrics should be employed, including both FR and NR approaches. What remains is the challenge of selecting appropriate metrics for different applications, a choice which has been approached in literature via IQA competitions and IQA datasets.

Since 2017 the Computer Vision and Pattern Recognition (CVPR) has hosted the New Trends in Image Restoration and Enhancement (NTIRE) workshop, in which several challenges are presented to the research community. Such challenges include super resolution, shadow removal and low light enhancement. The most recent challenge for IQA was held in 2022 [27], following on from the FR only competition held in 2021 [179]. In both competitions, the Perceptual Image Processing ALgorithms (PIPAL) dataset [165] was used to assess the competition entries. To assess the quality of the proposed metrics, their scores were compared against the MOS using a combination of the SRCC and PLCC to obtain a 'main score'. The winners of the two tracks were MANIQA [180] and AHIQ [181], who achieved the highest score for NR and FR metrics respectively. Since 2022, the challenge of single image IQA has not been renewed, however a challenge on video quality assessment was presented in 2023 [182], and the assessment of AI generated content was presented in 2024 [183].

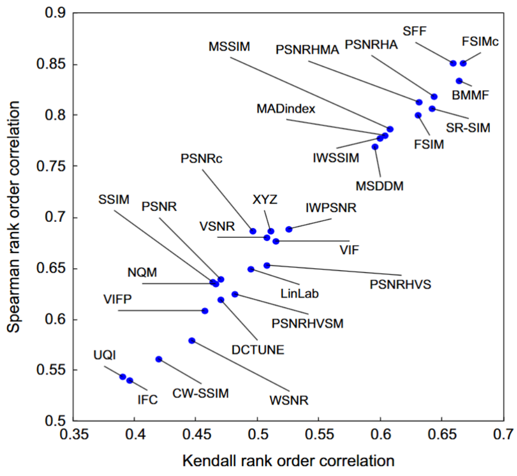
More recently, the European Conference on Computer Vision hosted the competition for Advances in Image Manipulation (AIM) [184] which focuses on NR IQA metrics for Ultra High Definition (UHD) images. This competition is facilitated by a dataset released from the same authors [185], in which a metric comparison is also performed. Consisting of 6,073 images, this dataset relies on naturally occurring distortions instead of synthetic. A panel of experts were shown these images and asked to rate them on a scale of 0% (bad) to 100% (excellent), therefore allowing a comparison between human perception and IQA metrics. Within the dataset paper, the QualiCLIP [186] metric is found to correlate the best with human perception. For the AIM competition, a combination of PLCC, SRCC and KRCC were utilised to identify which metric best aligned with the MOS. It was found that the top three competitors all outperformed QualiCLIP from the dataset paper, and the the overall winner was UIQA [187].

As well as competitions open to public entries, many works perform IQA comparisons via the creation of IQA datasets, with which, existing metrics can be compared with new metrics.

One such dataset is that of PIPAL from Gu et al. [165, 189], used in the NTIRE2022 competition for IQA. This dataset consists of 250 original images, degraded by 40 different distortion types, resulting in 29k total images. In ad-



(a) PIPAL [165].



(b) TID2013 [188].

Figure 3.10: Correlation results from TID2013 and PIPAL IQA datasets.

dition to traditional distortions, this dataset also contains output images from image recovery algorithms such as denoising and super-resolution, which include GAN-based methods. This is motivated by the fact that such image recovery algorithms are capable of generating complex textures, or unrealistic artifacts that metrics may be incapable of interpreting [179]. The MOS is obtained for each image via an Elo rating system, whereby observers are asked to choose between two images, and the chosen image's score is updated accordingly. Using this system, it is found that the PieAPP [190] metric provides the highest correlation, followed by WaDIQaM [191], where the correlation metrics used are that of SRCC and KRCC. The plot of metric correlations can be seen in Figure 3.10

Another IQA dataset is TID2013 from Ponomarenko et al. [188], which is a successor to TID2008 [192]. This dataset consists 25 original reference images, each of which is distorted by 24 different distortions, where there are 5 levels of each distortion. This resulted in a dataset of 3,000 distorted images, each of which having a MOS obtained from subjective opinions from human observers. Using these images, and their corresponding MOS, numerous IQA metrics were tested and their correlation was measured with SROCC and KROCC. It was found that the FSIMc [193] metric provided the best overall correlation, as seen in Figure 3.10.

By comparing the two graphs shown in Figure 3.10, it can be seen that the highest correlating metric on PIPAL (PieAPP) is far below that of the highest correlating metric on TID2103 (FSIMc). This observation does not imply that the PieAPP is an inferior metric, but rather that the PIPAL dataset is more challenging for the metrics to agree with the MOS scores. This therefore shows that the PIPAL dataset achieved its goal of presenting the metrics with a difficult task, therefore prompting the development of new metrics more specifically targeted toward outputs of image recovery algorithms.

Many other IQA datasets exist such as LIVE [194], LIVE In the Wild [195], CSIQ [196], Kadid 10k [166], KonIQ-10k [197]. Where the common theme is the acquisition of data either via real life distortions [195], or the post acquisition-application of such [166, 194, 196, 197]. These images are then passed to a series of observers, from which an MOS is obtained. Using this, new metrics are introduced, and compared with existing ones to identify which correlate positively with human observation.

Within the various works regarding the comparison of IQA techniques, a great number of metrics have been proposed. Traditionally, these rely on statistical features extracted from images. Lately however, deep learning has dominated the state-of-the-art for IQA, as it has for most image processing tasks. Such metrics can be found in both the NTIRE2020 and AIM competitions, where the winners, MANIQA, AHIQ and UIQA, all employ complex modern deep learning techniques such as vision transformers.

3.6.1 Metrics for Turbulence mitigation

Image quality assessment after turbulence mitigation is very similar to that in other image processing domains, where the majority of literature rely on PSNR an SSIM scores [60, 61, 76, 87, 198–202].

Other metrics that have been utilised include LPIPS [158, 203], NIQE [152, 204], BRISQE [144, 152], FID [158], BIQI [152] and PIQUE [144], where the two works from Liu et al. [144, 205] make use of multiple different metrics to assess the quality of their proposed turbulence mitigation algorithms, therefore supplying an in-depth analysis of their true performance.

These metrics are employed due to their proposed similarity to human perception, therefore promoting algorithms that produce perceptually pleasing images. In the case of post-mitigation automation however, it matters not if humans find an image perceptually pleasing. What matters is if the automated system can make good use of the generated images. The work from Zhang et al. [88] aim to address this by presenting a dataset of images affected by turbulence, where the image contents is text printed onto paper. Therefore, by performing text recognition before and after mitigation, an algorithm can be fully evaluated. This is therefore one of the tests for the UG^2+ competition on turbulence mitigation [154, 162]. Another example of application based metrics can be found in the work from Philippe et al. [206] who apply turbulence mitigation to better improve downstream facial recognition.

The work from Rai et al. [145] liberally apply the concept of algorithm quality assessment via post-mitigation applications. In their work, they test their proposed algorithm on multiple applications such as depth estimation, classification and keypoint identification. This concept is further highlighted in the work from Wang et al. [204] who state that the enhancement of downstream tasks is a more important goal than obtaining the best IQA metric scores. It would therefore be beneficial to identify a metric that correlates with such downstream tasks.

To attempt to identify the most suitable NR metric for turbulence mitigation applications, the work from Anantrasirichai et al. [57] make use of physically simulated turbulence data, induced with gas hobs. Using this simulation method, the ground truth is known, therefore allowing both NR and FR metric scores to be obtained. By comparing the correlation between the chosen NR metrics, and the corresponding FR scores, it was found that their own developed metric, QSVR, that best agreed with that of the FR scores.

In the work from Zhang et al. [88], an investigation into metrics for turbulence mitigation is also undertaken, where the metrics studied were PSNR, SSIM, LPIPS and CW-SSIM. As the work made use of their own simulation tool, the exact distortions present in the test images were known, therefore allowing the correlation between each metric and the level of turbulence to be obtained, where it was found that the CW-SSIM metric is more sensitive to image distortions due to turbulence. This finding is further supported by the work of Groff et al. [207], who, after comparing the various SSIM techniques, identified CW-SSIM as the most effective.

In line with identifying metrics that agree with turbulence levels, the work from Sharghi et al. [208] apply metrics to real turbulence, and aim find the metric that best correlates with the measured C_n^2 value. In this, they compare the NR metrics of BRISQUE, MTF and the Variance of Laplacian, where the MTF is measured using a known resolution chart placed in the images. It is found that the inverse of the Variance of Laplacian provides the best correlation with C_n^2.

As in the work from Anantrasirichai et al. [57], several works have aimed to develop a metrics specifically for turbulence mitigation applications.

One such metric from Saha et al. [163] makes use of edge detection in the captured images to present a Line Deviation Metric. It is proposed that high quality turbulence mitigation techniques should correct any warping distortions present. They judge such warping by performing edge detection followed by a hough line transform, from which they quantify the angular deviations from horizontal/vertical.

Another method of turbulence mitigation quality assessment could be that of attempting to estimate the level of turbulence in a given image. Therefore, if the level of turbulence is reported as low, the image will be of better quality. Such an investigation was undertaken by Saha et al. [62], who aimed to directly estimate the level of C_n^2 in an image using a neural network where, for training purposes, real turbulent images were acquired alongside the measurement of C_n^2.

Another turbulence specific metric is proposed by Hill et al. [209], who present 'Background-Var', which leverages the fact that the background of a given sequence should not move, especially if captured with a static camera. By segmenting this background and measuring the temporal variance, the quality of an image can be obtained, where a lower variance corresponds to a higher quality, stable image.

3.6.2 Investigated Metrics

The work in Chapter 6 presents a detailed investigation into image quality metrics for turbulence mitigation. In order to provide context to this investigation, the metrics used are first detailed here, where, due to the abundance of metrics developed in the IQA field, a full test of all available metrics would be infeasible. The metrics used are implemented using the PyIQA python package [210], which is an up to date library with implementations of many traditional and modern IQA techniques. The use of this library enables seamless implementation of the metrics and provides a common framework that facilitates valid comparisons in future research with consistent implementations

The list of chosen metrics is reported in Table 3.2, where it can firstly be seen that some metrics report 'highest best' (↑) scores, whilst others report 'lowest best' (↓), with the former indicating that higher scores reflect better image quality, and the latter indicating the opposite. It can also be seen that metrics are either based on traditional mathematical approaches or deep learning techniques, where the details of the base architectures were presented in Section 3.3.

As well as the details provided in Table 3.2, a brief description of each metric is now provided.

Table 3.2: List of metrics chosen for study. Showing minimum and maximum possible scores and the year of publication. The core methodologies used are also shown, being either data driven (deep learning), or mathematical. ↑ indicates that higher scores are better, whilst ↓ indicates that lower scores are better. Metric implementations are from the PyIQA library [210].

	Metric	**Range**		**Year**	**Base DL Architecture**	**Mathematical Approach**
		Min	**Max**			
NR	ARNIQA ↑ [211]	0	1	2024	SimCLR* [212]	
	BRISQUE ↓ [213]	0	100	2012		NSS → SVM
	CLIPIQA ↑ [214]	0	1	2023	CLIP [127]	
	LIQE ↑ [215]	1	5	2023	CLIP [127]	
	MANIQA ↑ [180]	0	1	2022	ViT [125]	
	NIQE ↓ [216]	0	100	2012		NSS → MVG
	PI ↓ [217]	0	100	2018		Ma et al.[218] and NIQE [216]
	PIQUE ↓ [219]	0	100	2015		MSCN coefficients
	Q-ALIGN ↑ [220]	1	5	2023	LLM†	
	QualiCLIP ↑ [186]	0	1	2024	CLIP [127]	
	TOPIQ_NR ↑ [174]	0	1	2024	ResNet-50 [114]	
FR	AHIQ ↑ [181]	0	1	2022	ResNet-50 [114], ViT [125]	
	CW-SSIM ↑ [221]	0	1	2009		Complex Wavelet Transform
	DISTS ↓ [222]	0	1	2020	VGG [120]	
	FSIM ↑ [193]	0	1	2011		PC and GM evaluation
	LPIPS ↓ [223]	0	1	2018	AlexNet [19]	
	MS-SSIM ↑ [224]	0	1	2003		Scaled Image statistics
	PieAPP ↓ [190]	0	5	2018	Novel DCNN	
	PSNR ↑	0	100	N/A		Log pixel comparison
	SSIM ↑ [171]	0	1	2004		Image statistics
	TOPIQ_FR ↑ [174]	0	1	2024	ResNet-50 [114]	

NSS: Natural Scene Statistics, **LMM**: Large Language Model
SVM: Support Vector Machine, **MVG**: Multivariate Gaussian
MSCN: Mean Subtracted Contrast Normalized
PC: Phase Congriguity, **GM**: Gradient Magnitude
*SimCLR makes use of the ResNet-50 architecture
†LLM model: mPLUG-Owl02 [225], makes use of ViT-L from [127]

No Reference Metrics

ARNIQA[211]**:** The leArning distoRtion maNifold for Image Quality Assessment metric makes use of the ResNet-50 [114] architecture and a 2-layer Multilayer Perceptron (MLP) to provide image quality scores. They adopt a self-supervision mechanism, whereby crops are extracted from two images and degraded equally. After being processed by the model, the loss function aims to maximise the similarity between the embeddings obtained from each crop, whilst also minimising the similarity between another pair of crops, degraded by a different distortion. This allows the model to prioritise the recognition of the underlying distortion, regardless of the image contents.

BRISQUE[213]**:** The Blind/Referenceless Image Spatial Quality Evaluator exploits Natural Scene Statistics by first calculating the Mean Subtracted Contrast Normalised (MSCN) coefficients of an image as,

$$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma(i,j) + C} \tag{3.19}$$

where i and j are spatial indices, C is a stability constant and the terms $\mu(i,j)$ and $\sigma(i,j)$ are defined as,

$$\mu(i,j) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l} I_{k,l}(i,j) \tag{3.20}$$

$$\sigma(i,j) = \sqrt{\sum_{k=-K}^{K} \sum_{l=-L}^{L} w_{k,l} \left(I_{k,l}(i,j) - \mu(i,j)\right)^2} \tag{3.21}$$

where $K = L = 3$ are the height and width of a local patch, $w_{k,l}$ is a Gaussian weighting function and $\mu(i,j)$ is the mean of the local patch. These MSCN coefficients are then fit to both the Generalised Gaussian Distribution (GGD) and the Asymmetric Generalised Gaussian Distribution (AGGD) to provide 36 features. A Support Vector Machine (SVM) is then trained using these features and the LIVE database [194]. The resulting SVM then represents the BRISQUE metric, where the same 36 features from an unseen image can be fed in to the pre-trained model to obtain a quality score.

CLIPIQA[214]**:** The CLIP Image Quality Assessment metric, as the name suggests, makes use of the CLIP vision-language model [127]. It employs CLIP by passing in each image alongside two prompts: 'good image' and 'bad image'. Given the two features from these prompts $\mathbf{t}_1$ and $\mathbf{t}_2$, the cosine similarity between them and the image embedding $\boldsymbol{x}$ can be evaluated as,

$$s_i = \frac{\boldsymbol{x} \odot \boldsymbol{t}_i}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{t}_i\|}, \quad i \in \{1, 2\}, \tag{3.22}$$

The Softmax function is then used to calculate a final score as,

$$\bar{s} = \frac{e^{s_1}}{e^{s_1} + e^{s_2}} \tag{3.23}$$

which therefore provides a score of how similar the image is to the prompt $\mathbf{t}_1$ ('good image'), so a higher score corresponds to a better quality image.

LIQE[215]**:** The Language-Image Quality Evaluator metric also make use of the CLIP vision-language model [127]. In this case, the metric first builds 495 different possible prompts, with the template "a photo of a(n) {s} with {d} artifacts, which is of {c} quality". Where there exist 5, 9 and 11 potential options for s, d and c respectively. Crops are then taken from each image and passed into the CLIP model, which provides visual and textual embeddings. The joint probability over s, d and c are then computed using cosine similarity. Following a marginalisation and discretisation step, a final image score is obtained.

MANIQA[180]**:** The winner of the 2022 NTIRE competition for NR IQA methods [27], the Multi-dimension Attention Network for IQA utilise a Vision Transformer (ViT) [125] to provide image quality scores, where the ViT is used to generate a series of output feature maps characterising each input patch. These feature maps are then passed into the proposed Transposed Attention Block (TAB), before being reshaped into a full image and fed to the Scale Swin Transformer Block (SSTB), which strengthens the local interaction between each patch. Finally a dual-branch structure is used to obtain a final image score.

NIQE[216]**:** Similarly to BRISQUE, the Natural Image Quality Evaluator first extracts the MSCN coefficients from the image, which is then partitioned into patches, selected via their local sharpness, calculated as

$$\delta(b) = \sum \sum_{(i,j)\in \text{ patch } b} \sigma(i, j) \tag{3.24}$$

where $\sigma(i, j)$ is the locally weighted standard deviation, calculated using Eq. (3.21). Patches that surpass a threshold are then selected for full characterisation, where, similarly to BRISQUE, a set of 36 features are extracted via the GGD and the AGGD. Unlike BRISQUE, NIQE then fits a Multivariate Gaussian Distribution (MVG) to these features (from multiple patches and images) to provide a natural model mean and covariance: μ_n and Σ_n. The quality of an unseen image can then be evaluated by fitting an MVG to the test image, providing μ_t and Σ_t. The NIQE score is then evaluated as,

$$\text{NIQE} = \sqrt{(\mu_t - \mu_n)^T \left(\frac{\Sigma_t + \Sigma_n}{2}\right)^{-1} (\mu_t - \mu_n)} \tag{3.25}$$

PI[217]**:** The Perception Index is calculated as the average of the NR metric from Ma et al. [218] and NIQE. It is defined as follows:

$$\text{PI} = \frac{1}{2}((10 - \text{Ma}) + \text{NIQE}) \tag{3.26}$$

where the metric from Ma et al. makes use of image statistics such as the Discrete Cosine Transform (DCT), wavelet decomposition and Principle Component Analysis (PCA) to train a regression model for image quality score prediction. Its output is a score between 0 (worst) to 10 (best).

PIQUE[219]**:** The Perception-based Image Quality Evaluator also makes use of the MSCN coefficients extracted from an image. These coefficients are segmented into non-overlapping blocks, where each block is either labelled as 'uniform' or 'spatially active', depending on the variance of the local MSCN coefficients. Each 'spatially active' block is then examined for Noticeable Distortion via edge analysis and Noise via analysis of the standard deviation. If found to have either distortion, that block is labelled as such. The final metric is then calculated as,

$$\text{PIQUE} = \frac{\text{Number of distorted blocks}}{\text{Total number of blocks}} \times 100 \tag{3.27}$$

Q-ALIGN[220]**:** This metric is the only Large Language Model (LLM) approach in this study. Similarly to CLIP, this architecture makes use of a Vision Transformer (ViT) to encode the input image. However, unlike CLIP, the resulting features are not compared with corresponding text encoded features, but are fed into the mPLUG-Owl02 LLM [225], which is fine-tuned from pretrained weights. Using a dataset with labels like 'The quality of the image is excellent', the LLM is trained to output similar text based responses. This text is then converted into a score between 1 (bad) and 5(excellent). This provides Q-ALIGN with more freedom for IQA, as the CLIP based architectures are dependant on the text encoding of binary antonyms such as ‘Good photo’ and ‘Bad Photo’.

QualiCLIP[186]**:** The highest ranking metric in the UHD dataset for AIM [185], the Quality-aware CLIP metric actively fine-tunes a pretrained CLIP model, as opposed to using the pre-trained model directly, to allow CLIP to become quality-aware. Firstly, overlapping crops are taken from each image such that the two crops have comparable quality, which are then degraded by 5 levels of distortion intensity. Each set of crops are then fed into the image encoder of CLIP whilst the text encoder receives two prompts: ‘good photo’ and ‘bad photo’. Loss functions are then applied to ensure that the more intense levels of distortion are appropriately tagged with ‘bad photo’, whilst less intense distortions are tagged with ‘good photo’. Each pair of crops from each distortion intensity are also examined, and a loss function ensures that they recieve similar scores, as they are only slightly different from one another due to their overlap.

TOPIQ_NR[174]**:** This network, referred to internally as CFANet, first extracts multi-scale features from the input image via ResNet-50 [114]. These features are then passed into the proposed Gated Local Pooling (GLP) layer, to standardise their spatial size. Self-attention (SA) blocks are then employed for further refinement of the features before they are processed progressively by Cross-Scale Attention (CSA) blocks. Finally, a pooling operation and MLP is used to define the final quality score.

Full Reference Metrics

AHIQ[181]**:** The winner of the 2022 NTIRE competition for FR IQA methods [27], the Attention-based Hybrid Image Quality Assessment Network utilises both a vision transformer and CNN in parallel. By providing each with an image pair: a distorted image and its corresponding reference, feature maps can be extracted from both. The feature maps from the ViT are then utilised to inform a deformable convolution step that is performed on the output features from the CNN (ResNet-50 [114]). The otuputs of this deformable convolution are then fused with the output of the ViT, and passed to a Patch-Prediction module. This module aims to perform spatial attention and patch-wise scoring to formulate a final image quality score.

CW-SSIM[221]**:** The Complex Wavelet SSIM measures the similarity of images by comparing phase patterns of local patches in the complex wavelet domain, where the patches are typically of size 3×3. It is evaluated as,

$$\text{CW-SSIM}(\mathbf{c}_x, \mathbf{c}_y) = \frac{2\sum_{i=1}^{N}|c_{x,i}||c_{y,i}| + k}{\sum_{i=1}^{N}|c_{x,i}|^2 + \sum_{i=1}^{N}|c_{y,i}|^2 + k} \cdot \frac{2\left|\sum_{i=1}^{N} c_{x,i}c_{y,i}^*\right| + k}{2\sum_{i=1}^{N}\left|c_{x,i}c_{y,i}^*\right| + k} \tag{3.28}$$

where N is the number of coefficients used in the local patch, $\mathbf{c}_x$ and $\mathbf{c}_y$ are the complex wavelet coefficients from x and y respectively, with $c_{x,i}$ and $c_{y,i}$ representing the i^{th} coefficients of the local patch. $c_{y,i}^*$ indicates the complex conjugate of $c_{y,i}$. As with SSIM, k is a small constant for the stabilisation of the calculation.

DISTS[222]**:** The Deep Image Structure and Texture Similarity Index make use of a modified VGG [120] to embed the distorted and ground truth images into 5 muliti-scale feature maps. For each scale, a difference score is evaluated that makes use of the feature maps mean and standard deviation. The sum of these difference across the 5 scales is then multiplied by two learnable weights to provide a final image quality score.

FSIM[193]**:** The Feature SIMilarity index evaluates perceptual image quality by combining phase congruency and gradient magnitude to model human visual sensitivity to structural and contrast information. It is evaluated as

$$\text{FSIM} = \frac{\sum_{x\in\Omega} \text{PC}_m(x) \cdot S_L(x)}{\sum_{x\in\Omega} \text{PC}_m(x)} \tag{3.29}$$

Here, Ω is the spatial area in question (i.e. the whole image, or a local patch). The term $\text{PC}_m(x)$ is the maximum phase congruency value between the reference and test images at pixel x. The local similarity measure $S_L(x)$ is defined as

$$S_L(x) = S_{\text{PC}}(x) \cdot S_{\text{GM}}(x)$$

where $S_{\text{PC}}(x)$ is the similarity of phase congruency and $S_{\text{GM}}(x)$ is the similarity of gradient magnitude.

LPIPS[223]**:** The Learned Perceptual Image Patch Similarity metric presents a framework of image quality assessment, that can make use of multiple classification models such as Alexnet [19], VGG [120] and Squeezenet [226]. From the chosen model, similarly to DISTS, multi-scale features are extracted, and scaled by a weighting vector. The L2 distance is then taken at each scale between the two feature sets. The average of these L2 values is then passed into a small secondary network that is trained to output a image quality score, given the average L2 distance.

MS-SSIM[224]**:** An extension of SSIM, the Multi-Scale SSIM computes image similarity across multiple spatial resolutions to better account for variations in viewing conditions and image scale. It is calculated as,

$$\text{MS-SSIM}(x,y) = [l_M(x,y)]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(x,y)]^{\beta_j} \cdot [s_j(x,y)]^{\gamma_j} \tag{3.30}$$

where M is the total number of scales (commonly $M = 5$) and $l_M(x,y)$ is the luminance comparison at the largest scale M. The terms $c_j(x,y)$ and $s_j(x,y)$ are the contrast and structure comparisons at scale j respectively. Finally the terms α_M, β_j, and γ_j are the exponents used to adjust the contribution of each component.

PieAPP[190]**:** This metric outperformed all others in the investigation performed by Gu et al. [189] when presenting the PIPAL dataset. It consists of a pairwise-learning framework that inputs two distorted images alongside the ground truth. An error score is obtained for each image using an encoder style CNN and two fully-connected layers. The two errors are then used to estimate the probability that one image is preferred over the other. Once trained, PieAPP is simply given the distorted and ground truth images in order to provide an image quality prediction.

PSNR: The Peak Signal-to-Noise Ratio calculates a logarithmic ratio between the maximum possible pixel value and the mean squared error between two images. It is calculated as,

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{L^2}{\text{MSE}}\right) \tag{3.31}$$

where MSE is the using the Mean Squared Error, defined as

$$\text{MSE} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - I'(i,j)]^2 \tag{3.32}$$

where L is dynamic range (i.e. 255 for 8 -bit images), I' is processed output image, I is the original image, and m and n are the image dimensions.

SSIM[171]**:** The Sturcutral SIMilarity index makes use of statistics extracted from the image to provide a quality score. It is calculated as,

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{3.33}$$

where μ_x and σ_x^2 are the mean and variance of x, and μ_y and σ_y^2 are the mean and variance of y respectively and σ_{xy} is the covariance between x and y. $c_1 = (k_1L)^2, c_2 = (k_2L)^2$ are small constants for the stabilisation of the division (typically $k_1 = 0.01$ and $k_2 = 0.03$). Finally, as with PSNR, L is the dynamic range.

TOPIQ_FR[174]**:** In order to utilise the TOPIQ metric for FR IQA, the same basic structure is used as was defined for the NR implementation. For FR IQA, ResNet-50 is used to extract multi-scale features from both images: distorted and ground truth. The Gated Local Pooling (GLP) layer then takes the feature maps from both images and outputs a single feature map. The remainder of the model then proceeds as before, with Self-attention blocks followed by Cross-Scale Attention. The final score is then found via pooling and a MLP.

3.7 Conclusion

The use of deep learning in modern machine vision tasks has become ubiquitous. The ability of these data-driven algorithms to gain an understanding of the problem at hand has led them to dominate state-of-the-art algorithms.

This chapter has provided a detailed description how such models have become so popular, and has provided details regarding their training procedure, from ANNs to CNNs. Due to the nature of this thesis, specific importance was placed on the description of CNNs and DL for image processing tasks. Detailing how a convolutional layer is employed and providing high level descriptions of popular architectures.

As well as a fundamental understanding of deep learning, an overview of the current literature was also provided, through the specific lens of turbulence mitigation. It was shown that, whilst sparsely reported at the beginning of this research, the application of deep learning has become the norm for turbulence mitigation.

An overview of the field of image quality assessment was also presented, where it is common for new metrics to be introduced alongside large datasets. Where the datasets can be used to perform accurate cross-comparison between different metrics, where no metric has an unfair advantage. This therefore allows the best metrics to be identified, and utilised in adjacent research.

When such datasets are used to find IQA metrics, it is common that each image is first provided a quality score via human observation, therefore providing a Mean Opinion Score (MOS) for each image. It is this MOS that the different metrics are compared with, where metrics that agree with human observation are deemed better. This comparison is known as correlation, where a metric that agrees with the MOS has a positive correlation. The various correlation metrics commonly used in literature were presented, alongside that of balanced accuracy. This is a metric that facilitates the optimisation of a classification threshold when presented with unbalanced data.

Finally presented in this chapter was a high level description of the metrics used throughout the thesis. The chosen metrics cover different approaches, from traditional metrics such as PNSR and SSIM, to more modern DL based approaches such as AHIQ and ARNIQA.

With the information presented in this chapter, the necessary background on deep learning and IQA metrics for the remainder of the thesis has been established. Especially in regard to the investigations presented in Chapters 5 and 6.

Chapter 4

Development of a Turbulence Simulator

4.1 Introduction

Fluctuations in the refractive index of air can cause light to divert from its intended path as it travels. When incident on a camera sensor, the consequences of these diversions are angle of arrival fluctuations and phase alterations. The resulting image is therefore degraded by spatially varied warping and blur. The processing of such an image can be rather challenging, as the combined warping and blurring result in a loss of image clarity and high frequency detail. Therefore, a common post-acquisition step is that of image recovery (i.e. turbulence mitigation) [12, 57, 227].

However, in order to design such mitigation algorithms, example data is needed. Where it is often the case where such post-acquisition processing methods require not only the turbulence affected images, but also the clean, turbulent free images as reference. This is especially true for the case of deep learning approaches, as a large quantity of paired data is necessary for training purposes.

In practice, it is extremely difficult to obtain such a dataset of turbulence affected imagery, as the high resolution, clean images, are unavailable. To obtain a suitable dataset, the atmospheric conditions during image capture would need to be controllable. Such as the work by Anantrasirichai et al. [57] where a turbulent atmosphere was induced with gas hobs, allowing high resolution images to be captured when the hobs were switched off. Such a physical simulation does not however model the real life outdoor turbulence over long distances, and the capture of such a dataset would be intractable.

To overcome this issue, a turbulent atmosphere can be simulated within software. Using such a simulation, in principle, allows any atmospheric conditions to be modelled, and repeated consistently to synthesise large datasets. In order to achieve such a simulation, a number of methods have been proposed. These methods can be separated into 3 dimensional (3D) and 2 dimensional (2D) approaches. 3D approaches [16, 79, 81], aim to fully simulate the propagation of light from one point to another, introducing the random distortions of turbulence during propagation. 2D [72, 76, 85] approaches aim to avoid the computational expense of 3D simulation and apply the effects of turbulence directly onto a 2D image. These 2D approaches are based on empirical data, from which they can extract suitable statistics. This allows the effects of turbulence to be reduced to a 2D representation which can then be applied to clean images [72, 85]. More recently, deep learning has also been utilised for turbulent data generation [84], where a Generative Adversarial Network (GAN) has been trained to generate new turbulence data, providing a method of data generation that takes a fraction of the time of a statistical model. The issue with such a model however is still that of training data, where the model first needs to understand what a turbulent image is, before being able to recreate the effects.

Of these methods, it is the 3D propagation model that is the most theoretically justifiable [77], and facilitates an accurate and controllable simulation. This chapter presents the development and implementation of such a simulator for generating anisoplanatic turbulent imagery. Based on the works of Schmidt [41] and Hardie et al. [81], the proposed simulator models a turbulent atmosphere as a series of discrete layers, each of which can be characterised by a single phase screen. Light from a point source is propagated through these screens from the source plane to the observation plane (camera sensor); the result of this is a complex matrix that can be transformed into a point spread function (PSF) using the lens transfer function defined in Eq. (2.57). Such a propagation represents a single optical path through the turbulent volume. By tracing a path from each point on the object plane, through the atmosphere to the image plane at the camera, a separate propagation can be performed for each. To achieve this, the phase screens are generated at an extended size where, once a trajectory (pencil ray) is traced through the atmosphere, its intersections with the extended phase screens are taken as centre points for a cropping operation. The cropped screens are then used for propagation. This inclusion of multiple optical paths is known as anisoplanatism (or anisoplantic) [5]. Once each pixel has been propagated through the atmosphere, each will have its own unique PSF, which can be applied to the image via spatially varied convolution. For the simulation of video sequences, the phase screens are expanded once more, and the video is simulated one frame at a time. In between each frame, the phase screens are translated laterally, simulating turbulent motion within a video, with each frame being temporally correlated to its neighbours.

To ensure that the final simulator is accurate, several validation steps are undertaken. This includes the validation of the propagation through a vacuum, as well as a turbulent atmosphere. The generated phase screens can be validated by comparing their structure functions to the theoretical values. Finally the PSFs generated by the simulator can also be compared against a theoretical ideal.

The simulator presented in this chapter was developed in MATLAB and is able to produce turbulent images as well as videos, where the speed and direction of the turbulence can also be defined. At the time of development, such a turbulence simulator was not publicly available, and any existing implementations were for internal use only. This simulation tool encompasses a full range of simulation techniques, from phase screen generation to image and video formation. The level of control over the final simulator is comprehensive, where not only the atmospheric conditions can be defined, but the speed and direction of turbulence between frames can also be chosen. Without such a simulation, the work achieved in Chapter 5 would not have been possible, as publicly available tools, such as the work from Mao et al. [76], had yet to be published.

Using this simulation implementation, three datasets were generated: 'turbulence images', 'turbulence videos' and 'turbulent imagenet'. The dataset 'turbulence images' consists of single images distorted by turbulence, and is separated into 4 sub-datasets, namely 'High', 'Medium', 'Low' and 'Mixed' turbulence, which facilitate a stepped difficulty testing approach for turbulence mitigation. Both 'turbulence videos' and 'turbulent imagenet' contain turbulent video sequences consisting of 15 frames each. Of these, 'turbulence videos' is the largest dataset, containing 148,884 total videos, whilst 'turbulent imagenet' contains 47,169. A key advantage of 'turbulent imagenet' is that, as it has been synthesised from the imagenet dataset, any turbulence mitigation algorithm can measure its success by any increase in post-mitigation image classification.

The code for the simulation tool is available upon request, whilst the 'turbulence videos' dataset is publicly available from the University of Strathclyde KnowledgeBase.

The remainder of the chapter is organised as follows. Section 4.2 first describes the layered atmosphere model, on which the simulator is based, as well as describing sampling constraints. Section 4.3 then presents the operations required for phase screen generation, from parameter calculation to screen synthesis. Section 4.4 details how the point source is created and propagated between screens. The final step of the simulator is then described in Section 4.5, where the expansion to anisoplanatic simulation is described, alongside PSF generation via the lens equation and the final image synthesis via spatially varied convolution. Section 4.6 then describes the steps taken to validate the simulator. Finally, Section 4.7 presents the three datasets that were synthesised for this work, and Section 4.8 and Section 4.9 provide the final discussion and conclusion.

4.2 Layered atmosphere model

As highlighted in Chapter 2, the propagation of a light source through a turbulent medium can be achieved with the split-step beam propagation method, which separates the full propagation distance into discrete layers. Such a layered model of the atmosphere is shown in Figure 4.1, where it can be seen that for this simulation, the entire propagation distance, L, is separated into $n-1$ layers, with n phase screens separated equidistantly between the source and observation planes. Each screen lies at a location z_n, and represents the atmospheric volume spanning the distance Δz_{n-1} (e.g. the volume between z_1 and z_2 is characterised by the phase screen at z_2). It is therefore worth highlighting the absence of a phase screen at location $z_n = L$. This is due to the fact that, during anisoplantic simulation (Section 4.5), the final phase screen is shared by all propagation paths. This would therefore result in all propagations undergoing identical phase alterations by the last screen, which would lead to excess tilt correlation [81]. By removing this final screen, this problem is alleviated, and its absence does not lead to a reduction in simulation accuracy.

Using the split-step propagation method, a point source can be propagated from the source plane (z_1) to each phase screen in turn until the observation plane is reached, where a PSF can be generated with the lens equation.

Sampling Constraints

In order to ensure an accurate simulation, sampling constraints are placed upon the discrete planes during propagation. The propagation volume has a width of $X = \Delta x N$, where Δx is the grid spacing and N is the number of samples.

Δx is calculated such that it is able to accurately represent each screen without

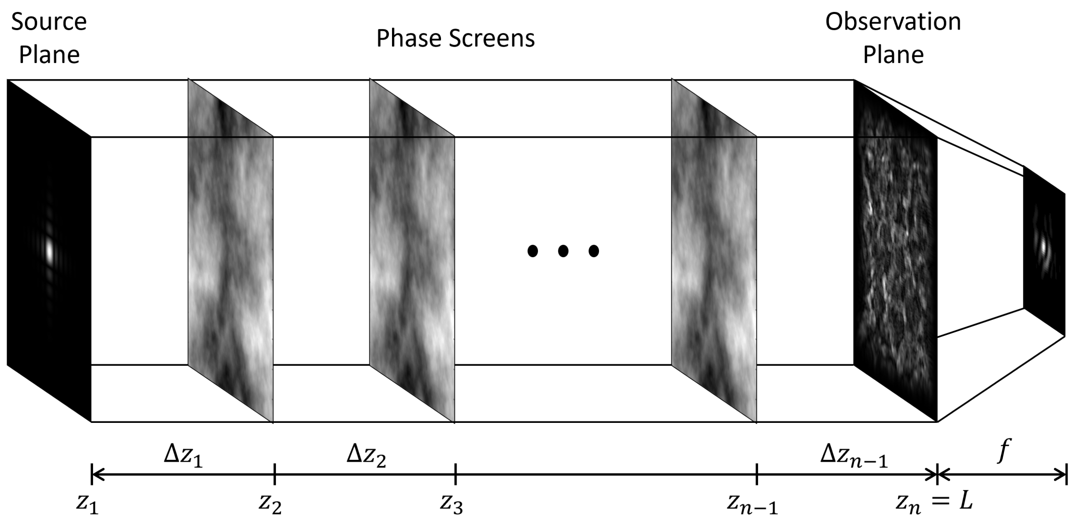


Figure 4.1: Volume representation of a single propagation. A point source at the source plane (z_1) is propagated through each phase screen ($z_2 \to z_{n-1}$) to the observation plane (z_n). A PSF is then generated via the lens equation over the distance f.

undersampling. To ensure this, Voelz's critical sampling rule is applied [228], as

$$\Delta x = \left(\frac{\lambda L}{N}\right)^{\frac{1}{2}} \tag{4.1}$$

where λ is the wavelength of the light. By rearranging this equation, the sample count N can be evaluated as

$$N = \frac{X^2}{\lambda L}. \tag{4.2}$$

These equations allow the sample count and spacing to be evaluated according to the desired volume width X. For the simulation of a camera system, X is related to the diameter D of the camera aperture by $X = sD$, where s is a scaling parameter. This scaling parameter is chosen such that the resulting screen width, X, is at least 4 times that of the aperture diameter (i.e $s \geq 4$), as well as ensuring that the resulting value of N is a power of two.

Therefore, once the diameter of the camera aperture is known, each sampling parameter can be evaluated in turn. This can be summarised by the following steps,

1. Choose s such that ($s \geq 4$)
2. Calculate the width of the propagation volume ($X = sD$)
3. Calculate the sample count N (as in Eq. (4.2))
4. Round N to the nearest power of 2

Table 4.1: User input parameters for turbulence simulation.

Symbol	Description	Category
C_n^2	refractive-index Structure Parameter	Turbulence Characteristics
l_0	Inner scale	
L_0	Outer scale	
λ	Wavelength of light	
Sub_Count	No. of Subharmonics	
$N_{subharm}$	Subharmonic grid size	
D	Camera aperture diameter	Propagation Environment
L	Propagation distance	
$nscr$	Number of phase screens	
$skip$	Pixel skip	
Image_Pixels	Output Frame size	Output Requirements
frame_count	Number of video frames to synthesise	
turbSpeed	How fast the turbulence moves between frames	
turbDirection	Which direction the turbulence moves between frames	

5. Calculate resulting grid spacing Δx (as in Eq. (4.1))

4.3 Turbulence simulation parameters

Control over the simulation is facilitated through the set of input parameters shown in Table 4.1. Whilst all parameters are listed here for reference, each will be explained in detail throughout the remainder of the chapter. These parameters can be grouped into three categories: Turbulence characteristics, Propagation Environment and Output Requirements.

4.3.1 Calculation of Fried Parameters

As defined in Chapter 2, the modified von Kármán Power Spectral Density (PSD) Function is defined as,

$$\Phi_{\phi_i}^{mvK}(f) = \frac{0.023\mathrm{e}^{-f^2/f_m^2}}{r_{0_i}^{5/3}\left(f^2+f_0^2\right)^{11/6}}, \tag{4.3}$$

where f is the frequency in cycles/m, $f_m = 5.92/2\pi l_0$ and $f_0 = 1/L_0$. Note that the PSD is now defined in terms of r_{0_i}, which facilitates the generation of each phase screens by its own corresponding Fried parameter. In order to find the Fried parameter for each screen, the behaviour of C_n^2 along the propagation path must first be studied.

This can be represented by the three different characteristics of turbulence, highlighted in Chapter 2, namely: the atmospheric coherence diameter (Fried parameter) (r_0), the isoplanatic angle (θ_0) and the log-amplitdue variance (σ_x^2). Each of which can be evaluated with respect to the C_n^2 value of each screen ($C_{n_i}^2$), and are computed using the discrete-sum versions of the originals as,

$$\hat{r}_{0,sw} = \left[0.423k^2 \sum_{i=1}^{n} \left(C_{n_i}^2\right) \left(\frac{z_{i+1}}{L}\right)^{\frac{5}{3}} \Delta z_i\right]^{-\frac{3}{5}} \tag{4.4}$$

$$\hat{\theta}_0 = \left[2.91k^2 L^{\frac{5}{3}} \sum_{i=1}^{n} \left(C_{n_i}^2\right) \left(1 - \frac{z_{i+1}}{L}\right)^{\frac{5}{3}} \Delta z_i\right]^{-3/5} \tag{4.5}$$

$$\widehat{\sigma}_{x,sw}^2 = 0.563k^{\frac{7}{6}} L^{\frac{5}{6}} \sum_{i=1}^{n} \left(C_{n_i}^2\right) \left(\frac{z_{i+1}}{L}\right)^{\frac{5}{6}} \left(1 - \frac{z_{i+1}}{L}\right)^{\frac{5}{6}} \Delta z_i \tag{4.6}$$

where it is worth noting the presence of $i+1$ for each position z. This is due to the fact that the source plane is located at z_1, meaning that the first screen, $C_{n_1}^2$, is located at position z_2, as shown in Figure 4.1.

Given that each of these parameters is expressed as weighted sums of C_n^2, a plot can further illustrate their importance at different locations along the propagation distance. This is shown in Figure 4.2. It can be seen that each parameter displays different characteristics at different points along the propagation path, with the Isoplanatic angle being most prevalent at the beginning of propagation, and the fried parameter coming more into effect nearer the observation plane. The log-amplitude variance is most apparent in the middle of propagation. By using all three of these parameters, the C_n^2 can be accurately represented along the entire propagation path, as shown by the dotted line.

The Fried parameter for each screen, with respect to $C_{n_i}^2$, is defined as

$$r_{0_i} = \left[0.423k^2 C_{n_i}^2 \Delta z_i\right]^{-\frac{3}{5}},$$

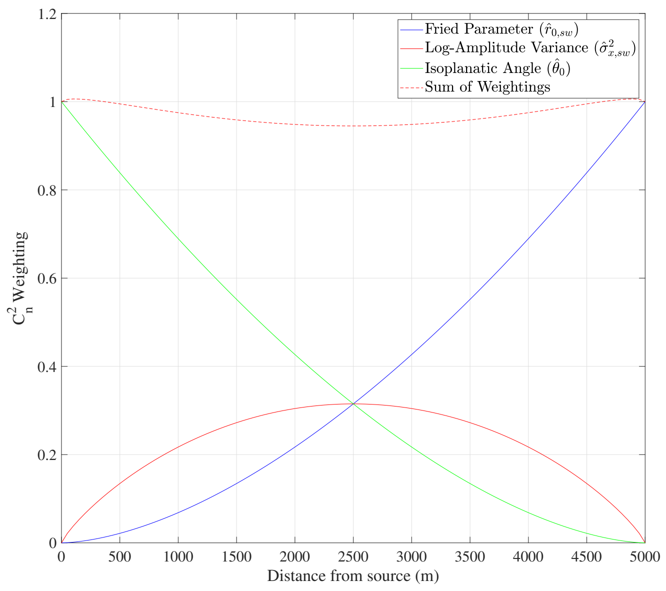


Figure 4.2: C_n^2 weighting of the Fried Parameter ($r_{0,sw}$), Isoplanatic Angle (θ_0) and Log-Amplitude Variance ($\sigma^2_{x,sw}$). The sum of which provide a consistent C_n^2 weighting over the entire optical path.

and can be rearranged to give

$$C^2_{n_i} = \frac{r_{0_i}^{-\frac{5}{3}}}{0.423k^2\Delta z_i}, \tag{4.7}$$

which can then be substituted into Eq.s (4.4), (4.5) and (4.6) to give,

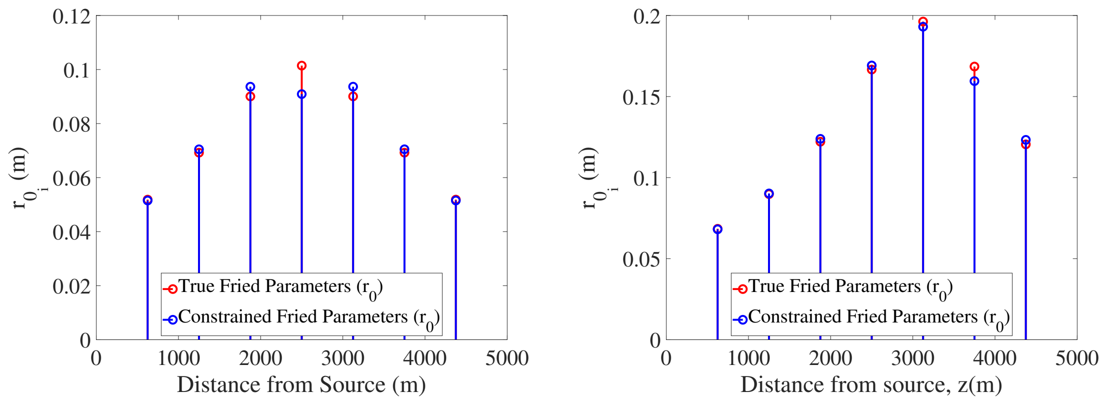
$$\hat{r}_{0,sw} = \left[\sum_{i=1}^{n} r_{0_i}^{-5/3}\left(\frac{z_{i+1}}{L}\right)^{\frac{5}{3}}\right]^{-\frac{3}{5}}$$

$$\widehat{\sigma}^2_{x,sw} = 1.311k^{-\frac{5}{6}}L^{\frac{5}{6}}\sum_{i=1}^{n} r_{0_i}^{-5/3}\left(\frac{z_{i+1}}{L}\right)^{\frac{5}{6}}\left(1-\frac{z_{i+1}}{L}\right)^{\frac{5}{6}}$$

$$\hat{\theta}_0 = \left[\sum_{i=1}^{n} L^{\frac{5}{3}}\left(1-\frac{z_{i+1}}{L}\right)^{\frac{5}{3}} 6.8794 r_{0_i}^{-5/3}\right]^{-3/5}$$

Using these equations, the values of r_0 for each screen can then be calculated by building a system of equations as,

$$\begin{bmatrix} \left(\hat{r}_{0,sw}\right)^{-5/3} \\ \frac{\hat{\sigma}^2_{x,sw}}{1.311k^{-5/6}L^{5/6}} \\ \frac{\left(\hat{\theta}_0\right)^{-5/3}}{6.8794L^{5/3}} \end{bmatrix} = \begin{bmatrix} \left(\frac{z_2}{L}\right)^{5/3} & \cdots & \left(\frac{z_{n+1}}{L}\right)^{5/3} \\ \left(\frac{z_2}{L}\right)^{\frac{5}{6}}\left(1-\frac{z_2}{L}\right)^{\frac{5}{6}} & \cdots & \left(\frac{z_{n+1}}{L}\right)^{\frac{5}{6}}\left(1-\frac{z_{n+1}}{L}\right)^{\frac{5}{6}} \\ \left(1-\frac{z_2}{L}\right)^{\frac{5}{3}} & \cdots & \left(1-\frac{z_{n+1}}{L}\right)^{\frac{5}{3}} \end{bmatrix} \begin{bmatrix} r_{0_1}^{-5/3} \\ \vdots \\ r_{0_n}^{-5/3} \end{bmatrix}. \tag{4.8}$$



(a) Fried parameters calculated for a consistent C_n^2 profile.

(b) Fried parameters calculated for a varied C_n^2 profile, with a higher C_n^2 at the source.

Figure 4.3: Resulting Fried parameters after optimisation of Eq. (4.8). Both constrained (blue) and unconstrained (red) values are illustrated, showing that too few phase screens can lead to suboptimal results.

This system of equations can then be solved according to two different constraints. Firstly, due to the presence of a negative power ($r_{0_1}^{-5/3}$), the resulting values of r_{0_i} must remain positive. Secondly, a constraint is placed upon the values of r_{0_i} such that the resulting log-amplitude variance, $\hat{\sigma}^2_{x,sw}$, for each screen does not exceed 20% of the total variance [81], i.e.

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} < \begin{bmatrix} r_{0_1}^{-5/3} \\ r_{0_2}^{-5/3} \\ \vdots \\ r_{0_n}^{-5/3} \end{bmatrix} < \begin{bmatrix} \frac{0.2\times\hat{\sigma}^2_{x,Sw}}{1.311k^{-5/6}L^{5/6}\left(\frac{z_2}{L}\right)^{\frac{5}{6}}\left(1-\frac{z_2}{L}\right)^{\frac{5}{6}}} \\ \frac{0.2\times\hat{\sigma}^2_{x,sw}}{1.311k^{-5/6}L^{5/6}\left(\frac{z_3}{L}\right)^{\frac{5}{6}}\left(1-\frac{z_3}{L}\right)^{\frac{5}{6}}} \\ \vdots \\ \frac{0.2\times\hat{\sigma}^2_{x,sw}}{1.311k^{-5/6}L^{5/6}\left(\frac{z_{n+1}}{L}\right)^{\frac{5}{6}}\left(1-\frac{z_{n+1}}{L}\right)^{\frac{5}{6}}} \end{bmatrix}.$$

This is done to ensure that phase fluctuations within any single screen remain within manageable limits and preserve simulation accuracy. If this constraint were to result in deviations from the ideal values of r_{0_i}, more atmospheric layers can be introduced, therefore allowing $\hat{\sigma}^2_{x,sw}$ to be distributed over more screens. Using these constraints, Eq. (4.8) can be evaluated to provide the final Fried parameters for each screen. Figure 4.3 illustrates two examples of Fried parameter calculation: one where C_n^2 remains consistent along the propagation path, and another where C_n^2 decreases from the source to the observation plane.

It can be seen in both cases that the constraints placed upon Eq. (4.8) result in deviations from the ideal r_{0_i} values which, as previously highlighed, can be overcome by the introduction of additional phase screens.

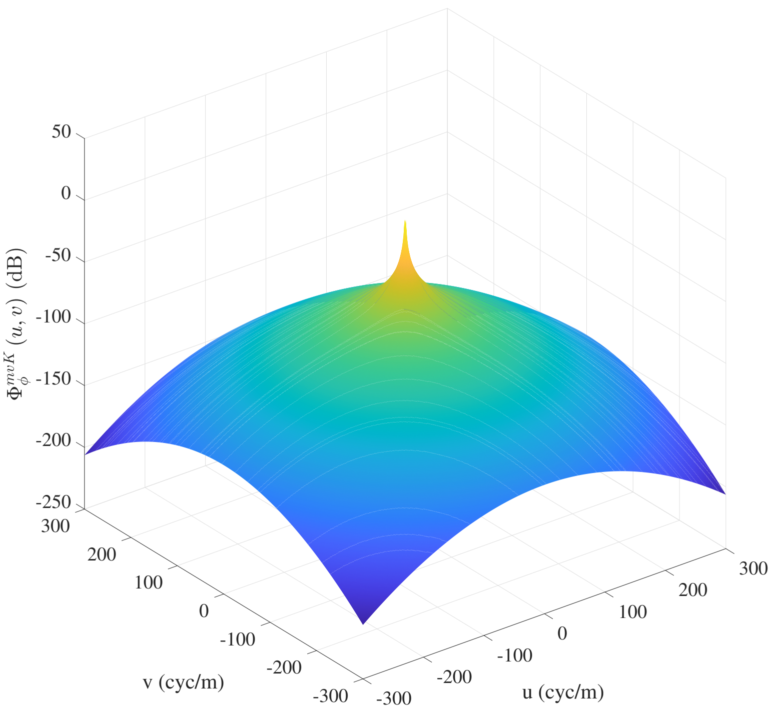


Figure 4.4: Modified von Kármán Power Spectral Density, Φ_{ϕ}^{mvK}, where $l_0 = 0.1m$, $L_0 = 300m$ and $r_{0,sw} = 0.1$.

4.3.2 Phase Screen Generation

Once the Fried parameters are found, they are used to evaluate the modified von Kármán PSD, $\Phi_{\phi_i}^{mvK}(f)$, for each screen (Eq. (4.3)). Its Fourier series coefficients, $c_{n,m}$, can then be obtained via square root (and rescaling) as,

$$c_{i_{n,m}} = \sqrt{\Phi_{\phi_i}^{mvK}\left(f_{x_n}, f_{y_m}\right)\Delta f}$$

where f_{x_n} and f_{y_m} are the spatial frequencies and $\Delta f = 1/(N\Delta x)$. Random realisations of these coefficients, $\hat{c}_{i_{n,m}}$, can then be found by multiplying by a zero mean, unit variance Gaussian random series as,

$$\hat{c}_{i_{n,m}} = \mathcal{N}\left(f_{x_n}, f_{y_m} \mid 0, 1\right) c_{i_{n,m}}$$

This multiplication allows for the random nature of turbulence to be simulated, resulting in no two phase screens being the same.

During simulation, a point source is propagated through each phase screen in turn, each of which imparts a turbulence induced optical phase change, $\phi(x, y)$. Assuming that this imparted phase is a Fourier transformable function, it can be

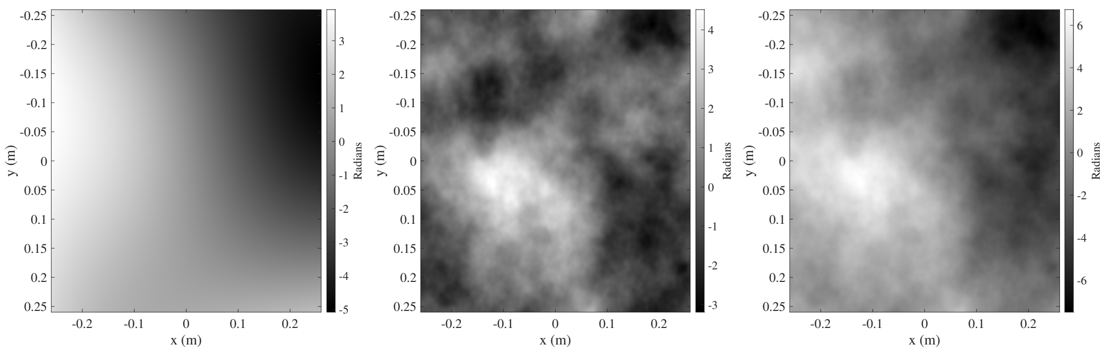


Figure 4.5: Three stages of phase screen construction. **Left**: Low frequency screen generated via subharmonics. **Middle**: Phase screen generated by sampling $\Phi_{\phi_i}^{mvK}$ at Δf. **Right**: The final phase screen, a combination of low and high frequencies.

written as a discrete Fourier series [41, 229] as,

$$\phi_i(x,y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \hat{c}_{i_{n,m}} e^{i2\pi(f_{x_n}x + f_{y_m}y)}$$

During propagation, the turbulence induced phase of each screen is then represented as,

$$\mathcal{T}\left[z_{i+1}\right] = e^{-i\phi_i(x,y)} \tag{4.9}$$

The limitation of this generation method is in the sampling of the modified von Kármán PSD at Δf. As shown in Figure 4.4, most of the power lies at the low spatial frequencies, which are inaccessible with a large Δf. The reduction of Δf would however increase Δx, which may lead to undersampling, as enforced by Eq. (4.1).

To access these frequencies, additional phase screens can be generated at subharmonics of Δf ($\Delta f_p = \Delta f/3^p$), which, when combined with the screen sampled at Δf, result in a phase screen that has suitably sampled the frequencies of the PSD. An example of the process of combining a low frequency subharmonic phase screen with a high frequency one can be seen in Figure 4.5.

4.4 Propagation

As derived in Chapter 2, the point source used for propagation is modelled as a 2D Gaussian windowed sinc function [41], defined as

$$U_{pt}\left(x,y\right) = \lambda L \alpha^2 e^{-i\frac{k}{2L}\left(x^2+y^2\right)} sinc\left[\alpha x, \alpha y\right] e^{-\frac{\alpha^2}{16}\left(x^2+y^2\right)}$$

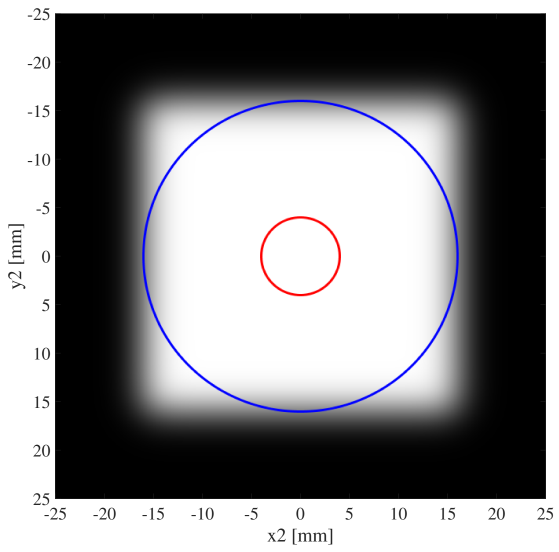


Figure 4.6: Resulting magnitude of a propagated point source through a vacuum, where $D = 8mm$ (Red Circle). The point source is designed such that the field at the observation plane is of a uniform amplitude with a width of $4D$ (Blue Circle).

where $\alpha = (4D)/(\lambda L)$. This source is designed such that, if propagated through a vacuum, the resulting field would be a $4D \times 4D$ patch of uniform amplitude on the observation plane, as is illustrated in Figure 4.6.

Propagation of the point source is performed using the Fresnel diffraction equation (also derived in Chapter 2), and repeated here for reference as,

$$\begin{aligned} U\left(\boldsymbol{r}_n\right) =& Q\left[\frac{m_{n-1}-1}{m_{n-1}\Delta z_{n-1}}, \boldsymbol{r}_n\right] \\ & \prod_{i=1}^{n-1}\left\{\mathcal{T}\left[z_{i+1}\right]\mathcal{F}^{-1}\left[\boldsymbol{f}_i, \frac{\boldsymbol{r}_{i+1}}{m_i}\right] Q_2\left[-\frac{\Delta z_i}{m_i}, \boldsymbol{f}_i\right]\mathcal{F}\left[\boldsymbol{r}_i, \boldsymbol{f}_i\right]\left(\frac{1}{m_i}\right)\right\} \\ & \left\{Q\left[\frac{1-m_1}{\Delta z}, \boldsymbol{r}_1\right] U\left(\boldsymbol{r}_1\right)\right\} \end{aligned} \tag{4.10}$$

where m is a scaling factor, defined as $m = \frac{\Delta z_2}{\Delta z_1}$, and allows for propagations between screens with different sample spacings. In the case of this simulation, all screens are sampled at Δx (i.e. $m = 1$). As highlighted in Chapter 2 this

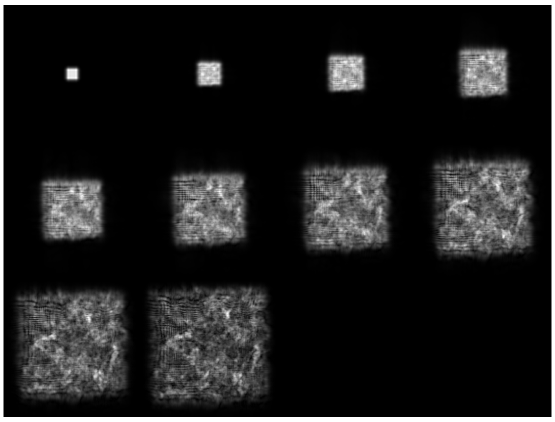


Figure 4.7: Magnitude of the propagating wave at each phase screen.

equation is written in operator notation, with following definitions.

$$Q[c,\boldsymbol{r}]\{U(\boldsymbol{r})\} = e^{i\frac{k}{2}c|\boldsymbol{r}|^2}U(\boldsymbol{r}) \tag{4.11}$$

$$Q_2[d,\boldsymbol{r}]\{U(\boldsymbol{r})\} = e^{i\pi^2\frac{2d}{k}|\boldsymbol{r}|^2}U(\boldsymbol{r}) \tag{4.12}$$

$$\mathcal{F}[\boldsymbol{r},\boldsymbol{f}]\{U(\boldsymbol{r})\} = \int_{-\infty}^{\infty} U(\boldsymbol{r})e^{-i2\pi\boldsymbol{f}\cdot\boldsymbol{r}}d\boldsymbol{r} \tag{4.13}$$

$$\mathcal{F}^{-1}[\boldsymbol{f},\boldsymbol{r}]\{U(\boldsymbol{f})\} = \int_{-\infty}^{\infty} U(\boldsymbol{f})e^{i2\pi\boldsymbol{f}\cdot\boldsymbol{r}}d\boldsymbol{f} \tag{4.14}$$

Another operation found in Eq. (4.10) is $\mathcal{T}\left[z_{i+1}\right]$, which represents the same as defined in Eq. (4.9), except now with the inclusion of a super gaussian boundary, g_{sg}. This boundary exists to attenuate any signal energy that reaches the border of the simulation during propagation, and is defined as,

$$g_{sg}(x,y) = e^{-\left(\frac{x}{\sigma\Delta x}\right)^n}e^{-\left(\frac{y}{\sigma\Delta x}\right)^n} \tag{4.15}$$

With this, the full propagation from source to observation planes can be performed, starting with the point source, $U\left(\boldsymbol{r}_1\right)$. After an initial multiplication with a quadratic phase factor, $n-1$ propagations are performed, in which the plane is first propagated along the distance Δz before being multiplied by the turbulence induced phase factor $\mathcal{T}\left[z_{i+1}\right]$. After these propagations have been completed, a final quadratic phase factor produces the final complex field, $U_0\left(x,y\right)$, situated at the observation plane $\left(z_n\right)$. The evolution of the waves magnitude as it passes through each screen can be seen in Figure 4.7.

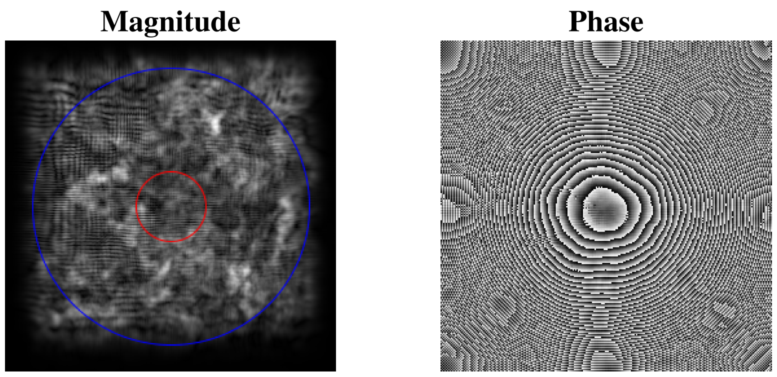


Figure 4.8: Magnitude and Phase of the final complex field, $U_0(x, y)$, as with propagation through a vacuum, the resulting magnitude has a width of $4D$ as intended.

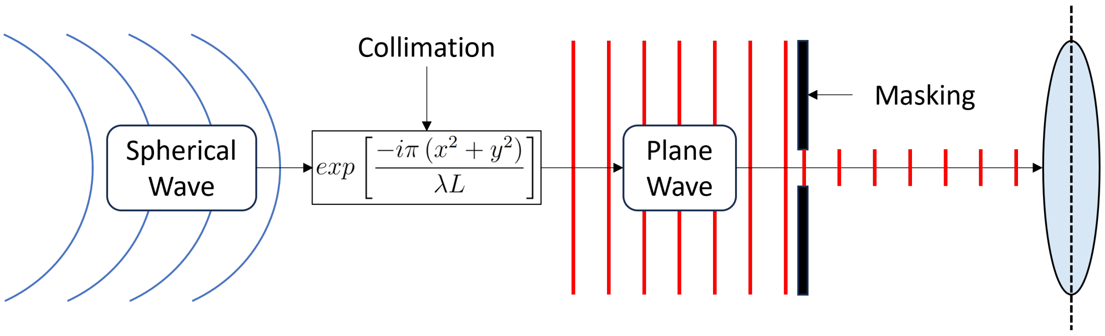


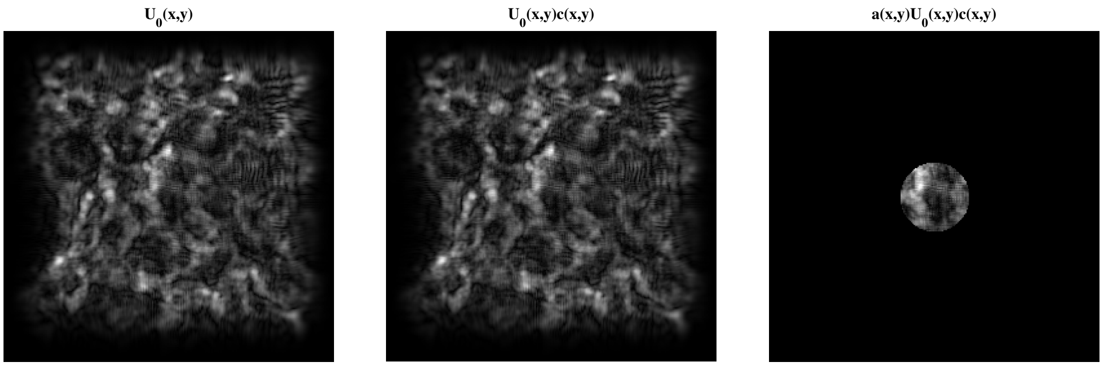
Figure 4.9: Steps taken to obtain the amplitude distribution before the application of the lens function. This consists of turning the incoming spherical wave into a plane wave via collimation, as well as a masking operation to restrict the wave to the aperture diameter D.

4.4.1 Lens Function

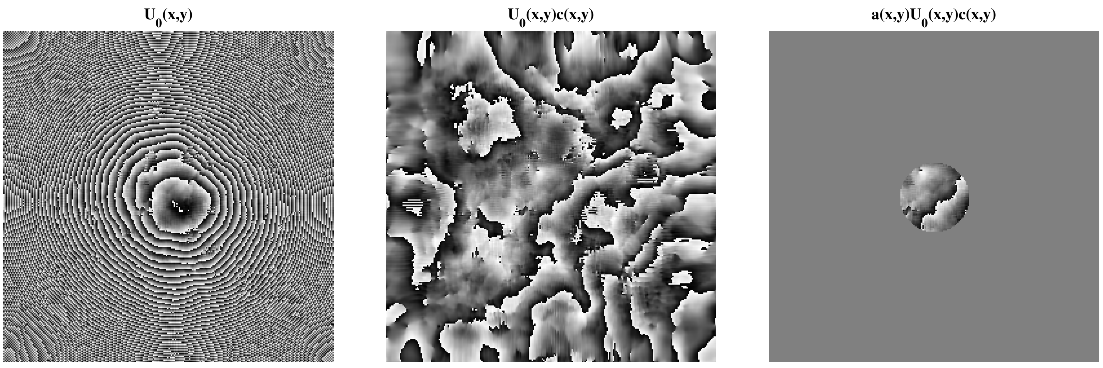
After propagation, the final field, $U_0(x, y)$, represents a spherical wave that is incident on the camera lens, the magnitude and phase of which is shown in Figure 4.8. In order to simulate the lens of the camera, this spherical wave first requires some additional processing, in the form of a collimation operation and masking [81]. These steps are illustrated in Figure 4.9.

It is shown that the collimation operation converts the incoming spherical wave into a plane wave such that it can be processed by the lens transfer function. The masking operation, $a(x, y)$, is then used to simulate the diameter of the camera lens, D, and simply suppresses any values outwith its effective aperture. The result of these two operations is a complex plane, $p(x, y)$, and can be defined mathematically as,

$$p(x, y) = a(x, y)\, U_0(x, y)\, exp\left[\frac{-i\pi(x^2+y^2)}{\lambda L}\right]. \tag{4.16}$$



(a) Magnitude progression.



(b) Phase progression.

Figure 4.10: Magnitude and Phase of the wave after the collimation and masking operations. The spherical nature of the phase is nullified by the collimation operation, whilst the magnitude is unaffected. The masking operation is applied to both magnitude and phase, and only allows the wave within the diameter D to remain.

An example of these operations on a complex screen is provided in Figure 4.10, where it can be seen that the magnitude is left unaffected by the collimation operation as expected, and both phase and magnitude are successfully masked as per the aperture diameter.

The output of Eq. (4.16) represents the amplitude distribution at the front of the lens (as shown in Figure 4.10). As derived in Chapter 2, the field intensity at the focal point of the camera, $h(x, y)$, is defined as,

$$h(x, y) = \left(|P(u, v)|^2\right)\Big|_{u=\frac{x}{\lambda l}, v=\frac{y}{\lambda l}}, \tag{4.17}$$

where $P(u, v) = \text{FT}\{p(x, y)\}$ and $\text{FT}\{\cdot\}$ denotes the forward Fourier transform. This resulting intensity field, $h(x, y)$, represents the PSF of the entire imaging system as a whole, from point source to focal plane. Its width, X_f, can be found via the mapping operation in Eq. (4.17) as,

$$X_f = X\lambda l = \frac{\lambda l}{\Delta x}$$

where X is the width of the propagated plane. When the final image is synthesised, the distance between each pixel is determined according to the Nyquist sampling rate [81], and is defined as

$$\delta_o = \frac{\lambda L}{2D}. \tag{4.18}$$

When this is projected through the lens onto the focal plane, the Nyquist sample rate becomes

$$\delta_f = \frac{\lambda l}{2D}. \tag{4.19}$$

This then allows the number of Nyquist samples/pixels in the PSF to be evaluated as,

$$N_f = \frac{X_f}{\delta_f} = \frac{2D}{\Delta x},$$

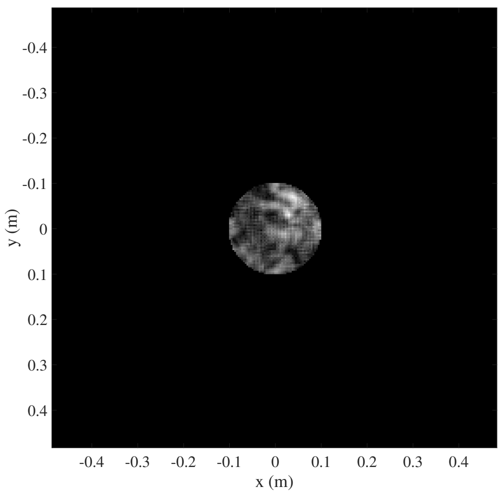
which can be written with respect to the propagated sample count N as,

$$N_f = \frac{2D}{\sqrt{\lambda L}}\sqrt{N},$$

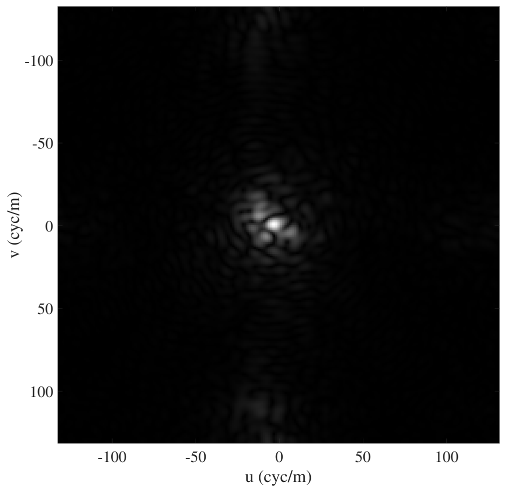
Due to the fact that $\frac{2D}{\sqrt{\lambda L}} < \sqrt{N}$, the value of N_f will always be smaller than N, therefore requiring a downsample from the original sample count, such that the values at each location of the PSF align with their corresponding pixels in the image. Note that this is not dependant on the focal length l, and is the reason for its absence in the simulation input parameters. In practice, the values of δ_f and X_f are only needed when displaying the spatial axes of the PSF. A final operation is the normalisation of the PSF to have a sum of 1 [81]. The stages of PSF generation are shown in Figure 4.11, where the spatial axes of the focal plane and nyquist PSFs are shown with a example focal length of 1.2.

4.5 Extension to Anisoplanatic Simulation

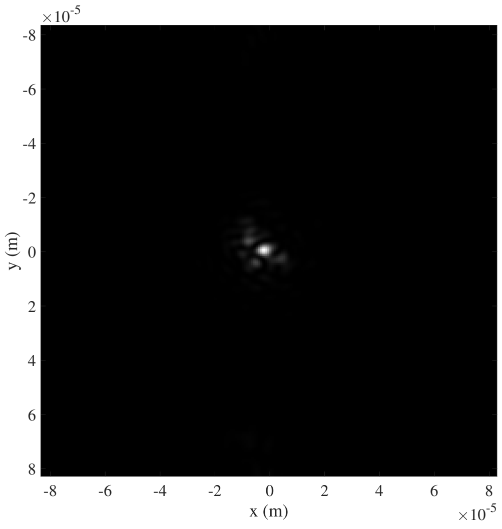
The process detailed up to this point has only taken into consideration a single propagation, producing a single PSF. If this was to be applied to each pixel in the image, the result would be an example of isoplanatic simulation, where the resulting distortion would be spatially invariant, and would assume that each source



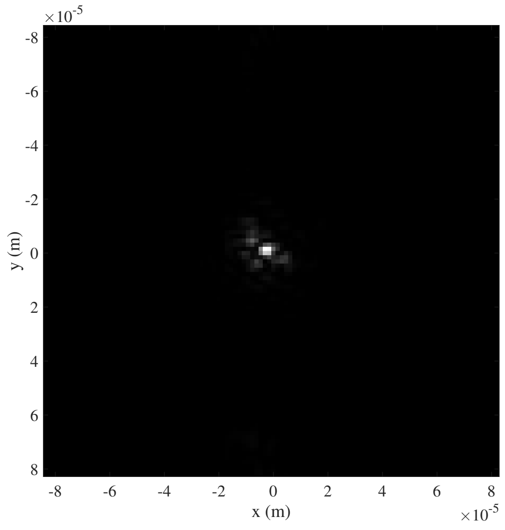
(a) $p(x,y)$



(b) $|P\,(u,v)\,| = |\mathrm{FT}\{p(x,y)\}|$



(c) $h(x,y) = \left(|P(u,v)|^2\right)\big|_{u=\frac{x}{\lambda l},v=\frac{y}{\lambda l}}$



(d) PSF sampled with $N_f = \frac{2D}{\Delta x}$

Figure 4.11: Stages of Point Spread Function generation. After masking, the PSF magnitude is obtained via Fourier Transform. The intensity is then calculated and the coordinates are remapped according to the focal plane. Finally, the PSF is resampled according to the Nyquist sampling rate. For this example, all spatial axes are shown for a camera with a focal length of 1.2.

of light (pixel) passed through the exact same volume of turbulent atmosphere.

A more realistic model would assign a unique optical path to each pixel, and perform a propagation for each. In this, the resulting simulation would take into account multiple angles of arrival, i.e. an anisoplanatic simulation [5].

Figure 4.12 provides an illustration of how such a simulation is performed, where the phase screens are expanded such that they represent a larger area of the turbulent atmosphere. For each pixel, a 'pencil ray' can then be traced from the source to the observation plane. The points at which these pencil rays inter-

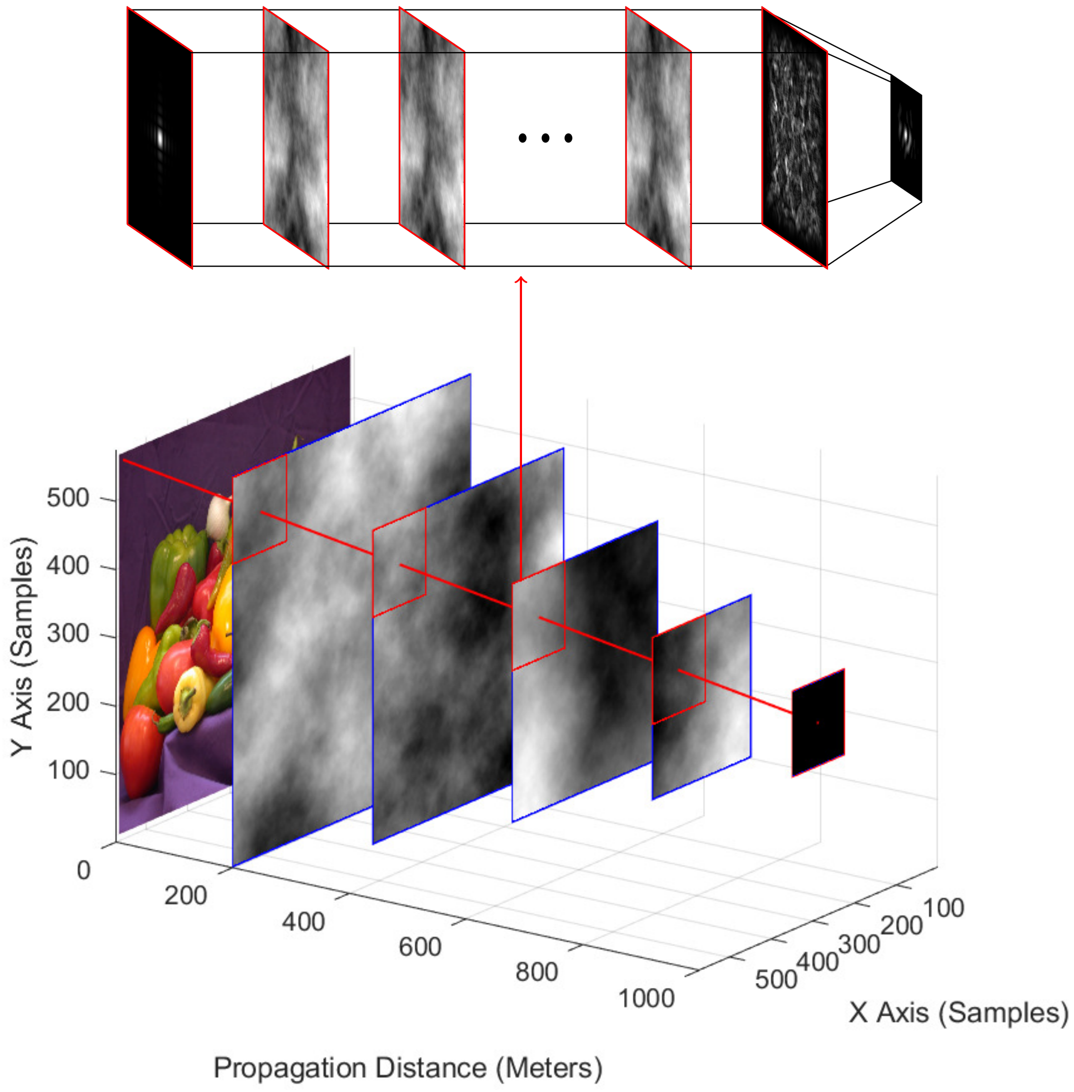


Figure 4.12: Geometry of 3D space in which phase screens are situated. For each pixel, a pencil ray is traced towards the observation plane. The intersections of the pencil rays correspond to the centre points for the cropped screens used for propagation (shown by a red border). The resulting single propagation makes use of these cropped screens to generate a single PSF for that pixel.

sect with the extended phase screens represent the centre points for a cropping operation. Each of the cropped phase screens is then used for a single propagation for that pixel (as described in Section 4.4), resulting in a unique PSF for its optical path. This process is then repeated for each pixel in the image, providing an array of PSFs that represent spatially varying warp and blur. In Figure 4.12 it can also be observed that if a phase screen were to exist on the observation plane, the final cropped phase screen for each pencil ray would be identical. As mentioned in Section 4.2, this would lead to excess tilt correlation and is the reason why there is no screen present at the observation plane.

In order to generate the phase screens at an extended size, their width, $\tilde{X}_i$, can be expressed as,

$$\tilde{X}_i = \Delta x \tilde{N}_i \tag{4.20}$$

where Δx remains unchanged from the single propagation calculations (Eq. (4.1)) and $\tilde{N}_i$ is the number of samples required at position i along the propagation path. As shown in Figure 4.12, the extended screen closest to the observation plane is smaller due to the convergence of the pencil rays.

As shown in Eq. (4.18), the grid of pixels at the source plane is sampled according to Nyquist [81] and provides an image width of

$$X_s = \delta_o N_s \tag{4.21}$$

where the number of samples, N_s, is determined by the input parameter 'Image_Pixels' (see Table 4.1). The largest screen (i.e. closest to the source plane) is sampled such that it is N samples larger than the source image for the accommodation of the corner pencil rays.

When generating the phase screens in Section 4.3, the screens were sampled at a frequency of $\Delta f = \frac{1}{N\Delta x}$. However, due to their increased size for anisoplanatic simulation, the sample frequency is instead calculated as

$$\Delta f = \frac{1}{\tilde{N}\Delta x}. \tag{4.22}$$

This allows the cropped screens to maintain the appropriate structure for a valid simulation.

A performance gain for the simulation process is possible at this stage. In fact, due to the nature of the pencil rays, each pixel will have a very similar path through the atmosphere as its immediate neighbours. If the angle of propagation between two paths is within that of the isoplanatic angle θ_0, it can be said that the resulting PSFs will be approximately the same [81]. Therefore, a skip parameter allows the pencil rays to be traced using a sparser grid of pixels. The PSFs of the remaining pixels are then estimated using bilinear interpolation. Figure 4.13 shows examples of these spatially varied PSFs.

Once each PSF is obtained, they can be applied to their corresponding pixel in the source image. This is achieved via spatially varied convolution, described as

$$y[m,n] = \sum_j \sum_i x[i,j] \cdot h_{m,n}[m-i, n-j] \tag{4.23}$$

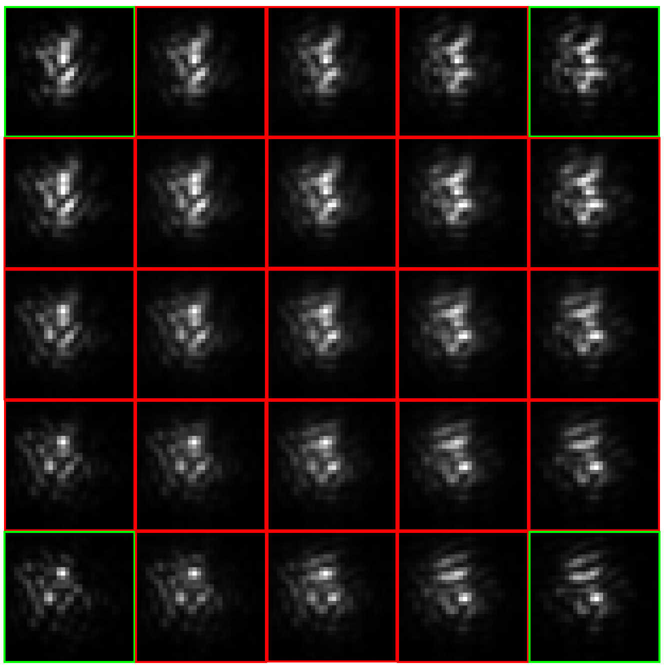
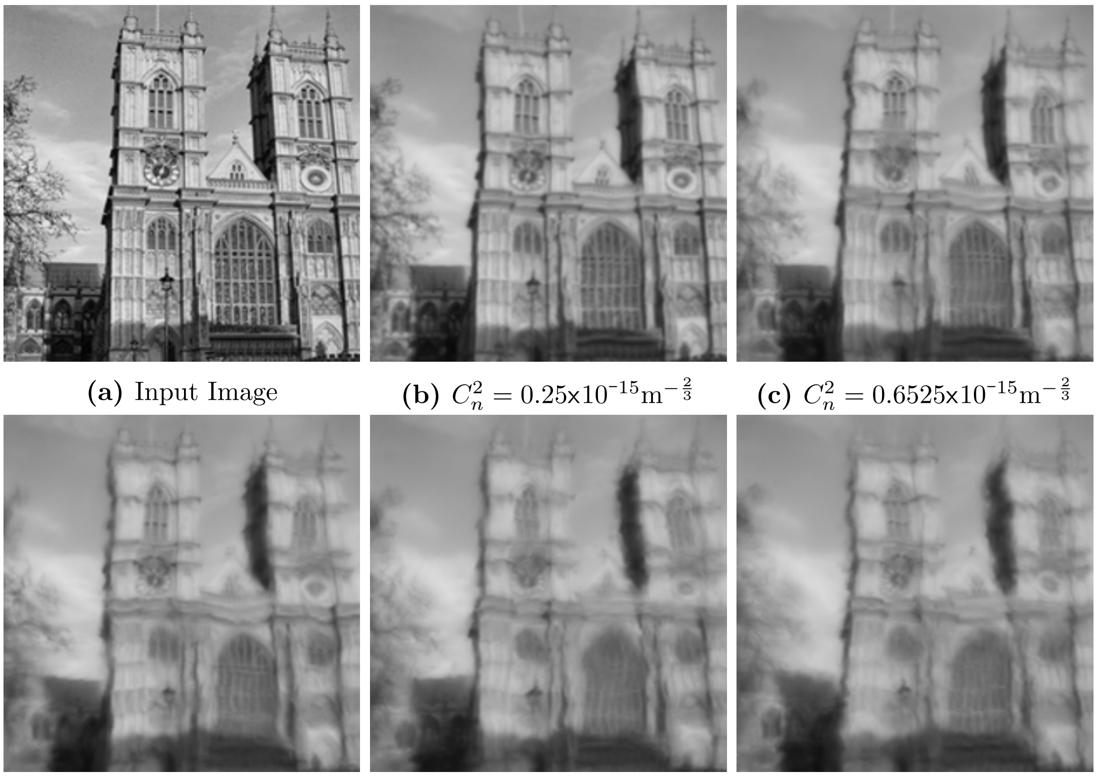


Figure 4.13: Example array of Point Spread Functions. PSFs with a green border are found via propagation, whilst PSFs with a red border are found via interpolation. Such interpolation decreases the computational complexity of image simulation. Due to similar paths taken by adjacent pixels, this computational saving does not impact simulation accuracy.

where y is the distorted output image, x is the clean input image and $h_{m,n}$ is the PSF associated with the pixel at location m, n.

In order to reduce the computations required for a single image or frame, it was decided that the simulation would only be applicable to grayscale images as, if an RGB image was required, three separate propagations would be necessary. It was deemed acceptable to restrict the simulations to grayscale due to the fact that the additional channels do not provide further information regarding the spatial structure of an image. A further issue with RGB simulation is the fact that the simulator is reliant upon the input wavelength λ. In its current state, a change in the value of λ would also require the re-generation of the phase screens. Due to the random nature of this generation, this would result in the propagation for each channel using a different random realisation of the phase screens. This would therefore result in three uncorrelated PSFs for each pixel which, when applied to a clean image, would result in chromatic aberrations as the different colours have tilted in different directions.

The simulation of colour images could be achieved, as in the work by Mao et al. [76], by assuming that the wavelength of light does not drastically affect the resulting PSF. Under this assumption, the wavelength of light can be chosen as a



(a) Input Image **(b)** $C_n^2 = 0.25\text{x}10^{-15}\text{m}^{-\frac{2}{3}}$ **(c)** $C_n^2 = 0.6525\text{x}10^{-15}\text{m}^{-\frac{2}{3}}$

(d) $C_n^2 = 0.875\text{x}10^{-15}\text{m}^{-\frac{2}{3}}$ **(e)** $C_n^2 = 1.1875\text{x}10^{-15}\text{m}^{-\frac{2}{3}}$ **(f)** $C_n^2 = 1.5\text{x}10^{-15}\text{m}^{-\frac{2}{3}}$

Figure 4.14: Example outputs from the developed simulator. Shown are five different levels of turbulence strenght (C_n^2), as well as the original ground truth image.

central value on the visible spectrum (e.g. $525nm$). A single simulation can then be performed, with the resulting PSF being applied to all three colour channels.

The result of the spatially varied convolution is a distorted image that represents an accurate depiction of turbulent distortion, examples of which can be seen in Figure 4.14.

A key observation to be made at this point is the assumption that the contents of the input image x all lie on the same plane. In reality, objects in a scene generally occupy different depths relative to the camera at the time of capture. Therefore, a fully accurate simulation would take this added depth information into account. i.e. for each pixel, the propagation distance should be bespoke. This however is impractical, as it would require knowledge of the depth of each pixel within the original image, as well as require propagation calculations for each pixel within the image. It is therefore assumed during simulation that the object plane is sufficiently far away from the observation plane such that any small differences in the original image depth are negligible compared with the overall propagation distance. Under this assumption, a single effective propagation distance provides an adequate approximation.

Finally, the simulation framework can be altered once more in order to facili-

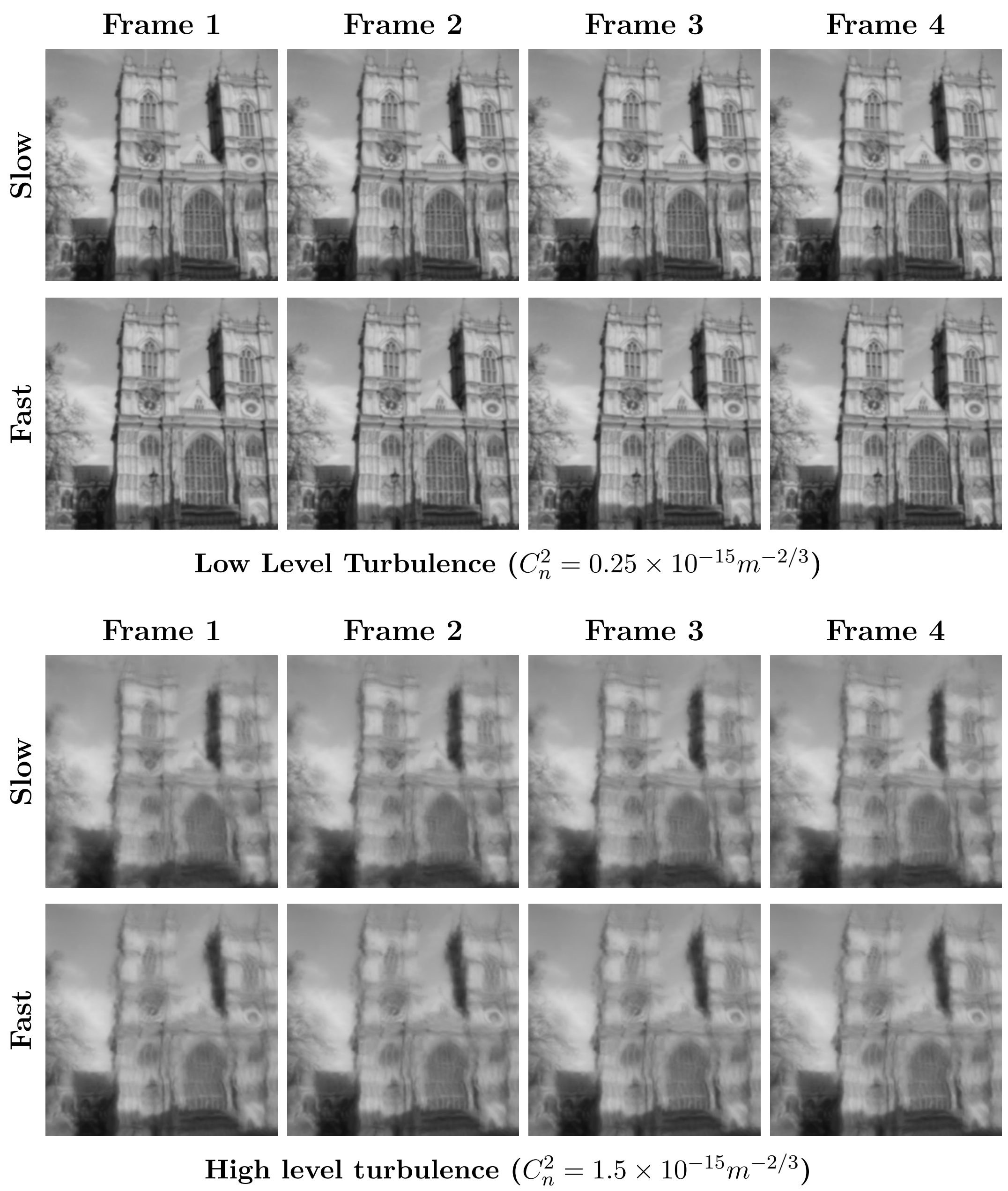

Frame 1
Frame 2
Frame 3
Frame 4
Slow
Fast
Low Level Turbulence ($C_n^2 = 0.25 \times 10^{-15} m^{-2/3}$)
Frame 1
Frame 2
Frame 3
Frame 4
Slow
Fast
High level turbulence ($C_n^2 = 1.5 \times 10^{-15} m^{-2/3}$)

Figure 4.15: Example video outputs from the developed simulator. Shown are the first 4 frames of four different example videos. Each was synthesised with either 'Fast' (turbSpeed=4) or 'Slow' (turbSpeed=1) turbulence. Two different levels of turbulence strength are shown.

tate the synthesis of turbulent videos, where the number of frames in the output video is determined by the 'frame_count' parameter (see Table 4.1). Once the number of frames is known, the phase screens are extended to an even larger size. After the first frame is synthesised, the screens are then translated on their 2D plane, before a second frame is synthesised, therefore simulating the temporally correlated movement of turbulence between frames. The direction and speed of this translation operation is determined by the 'turbSpeed' and 'turbDirection' input parameters, where 'turbSpeed' refers to the number of grid samples the screens are translated and 'turbDirection' is the direction of this translationm specified using any of the cardinal directions or diagonals (i.e.$\rightarrow\searrow\downarrow\swarrow\leftarrow\nwarrow\uparrow\nearrow$). The use of the 'turbSpeed' parameter can be seen in Figure 4.15, where the videos were either set to simulate 'Slow' (turbSpeed=1) or 'Fast' (turbSpeed=4) turbulence. These two cases are shown for both low ($C_n^2 = 0.25 \times 10^{-15} m^{-2/3}$) and high ($C_n^2 = 1.5 \times 10^{-15} m^{-2/3}$) turbulence (More examples, showing all 15 frames, can be found in Appendix B).

Given the ability to simulate temporally correlated images, the simulator can then be used to generate two types of video sequence: static or dynamic. Where for static video synthesis, the input is a single image, and for dynamic videos, the input is itself a video sequence. In static video generation, it is only the distortion due to turbulence that changes between each frame, whereas in the dynamic case, the contents of the video may move as well. This movement may originate from objects moving within the scene (a moving car) or from the camera itself moving (on a moving platform). The two types of video sequence pose two different problems for turbulence mitigation. In the static case, the goal is to simply recover the underlying high quality image, whereas for dynamic scenes, the turbulence mitigation algorithm is also required to maintain the temporal consistency between frames. It can be seen therefore that a turbulence mitigation algorithm designed for static scenes would not be applicable to dynamic scenes, as it would have no knowledge of how to process movement between frames that is not due to turbulence.

4.6 Validation

Before the simulator could be utilised to synthesise turbulent data, it first required validation to ensure its accuracy. This was done by first validating individual steps such as propagation and phase screen generation. The whole simulation could then be validated by comparing simulated point spread functions with their theoretical counterparts.

4.6.1 Propagation

The first step in the validation of the propagation step has already been presented in Section 4.4, where it was shown that the field found at the observation plane was a uniform field with a width of $4D$, as intended by the designed point source. This was shown to be true for both vacuum and turbulent propagation. Another validation step that can be performed is the analysis of the wave power as it is propagated through the phase screens.

The intensity of a wave is defined as the squared magnitude of the complex representation [54],

$$I_{i,j} = |A_{i,j}|^2$$

allowing the power of the wave to be calculated as,

$$P = \Delta x^2 \sum_{i,j} |A_{i,j}|^2 \tag{4.24}$$

where the use of Δx^2 accounts for the spatial scaling. By evaluating the wave power after each propagation step, we can see if any power is being lost during the simulation. Figure 4.16 shows how the wave power changes over both vacuum and turbulent propagations.

It can be seen that over the whole propagation distance, the power does not drop significantly from the original power supplied by the point source. The only reduction in power occurs near the end of the propagation, and is more apparent during a turbulent propagation. This is simply due to the super Gaussian boundary (Eq. (4.15)), and is only affecting the very edge of the propagating wave. The turbulent propagation is impacted more significantly due to the random spread of the propagating wave, resulting in more values being suppressed. Due to the masking operation detailed in Section 4.4, this loss in wave power does not affect the final simulation.

4.6.2 Phase Screens

The phase screens can be validated by comparing them with the theoretical wave structure function for a plane-wave source with Kolmogorov turbulence

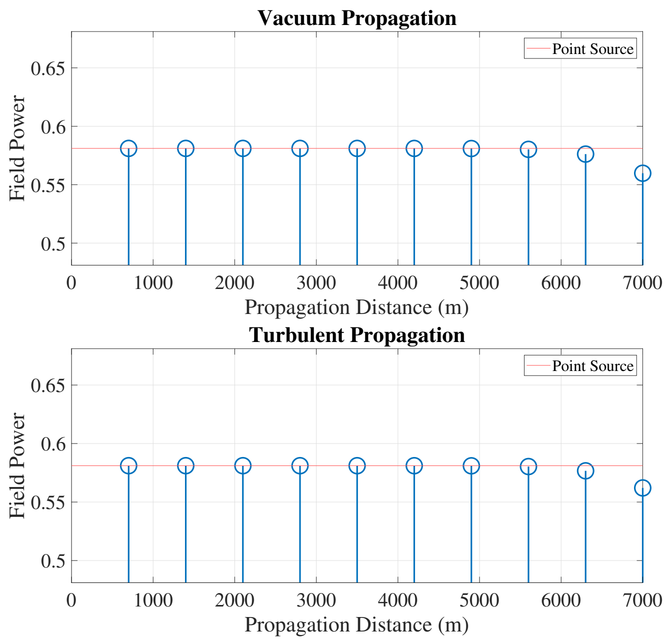


Figure 4.16: Power of the propagated wave at each step along the propagation path, alongside the original signal power from the point source. Shown are cases for both vacuum propagation (top) and turbulent propagation (bottom). The approximately constant signal power shows that the simulation is successfully performing split-step propagation.

[41], which is defined as,

$$D^K(|\Delta\mathbf{r}|) = 6.88\left(\frac{r}{r_0}\right)^{5/3} \tag{4.25}$$

where $r = \sqrt{x^2+y^2}$, r_0 is the atmospheric coherence diameter and $\Delta\mathbf{r}$ is the radial separation between pixels. The evaluation of a discrete 2D structure function is defined as [41],

$$D_{\phi'}(\Delta\mathbf{r}) = 2\mathcal{F}\left\{\mathrm{Re}\left[S\left(\mathbf{f}_1\right)A^*\left(\mathbf{f}_1\right)\right] - \left|P'\left(\mathbf{f}_1\right)\right|^2\right\} \tag{4.26}$$

where $\mathcal{F}\left\{\cdot\right\}$ is the Fourier transform and the values of $A(\mathbf{f})$, $P'(\mathbf{f})$ and $S(\mathbf{f})$ are defined as,

$$\begin{aligned} A(\mathbf{f}) &= \mathcal{F}\{a(\mathbf{r})\} \\ P'(\mathbf{f}) &= \mathcal{F}\left\{\phi'(\mathbf{r})\right\} \\ S(\mathbf{f}) &= \mathcal{F}\left\{\left[\phi'(\mathbf{r})\right]^2\right\} \end{aligned} \tag{4.27}$$

where $a(\mathbf{r})$ is the camera aperture aperture and $\phi'(\mathbf{r}) = \phi(\mathbf{r})a(\mathbf{r})$ is the masked phase screen.

A comparison between the theoretical structure function and the structure

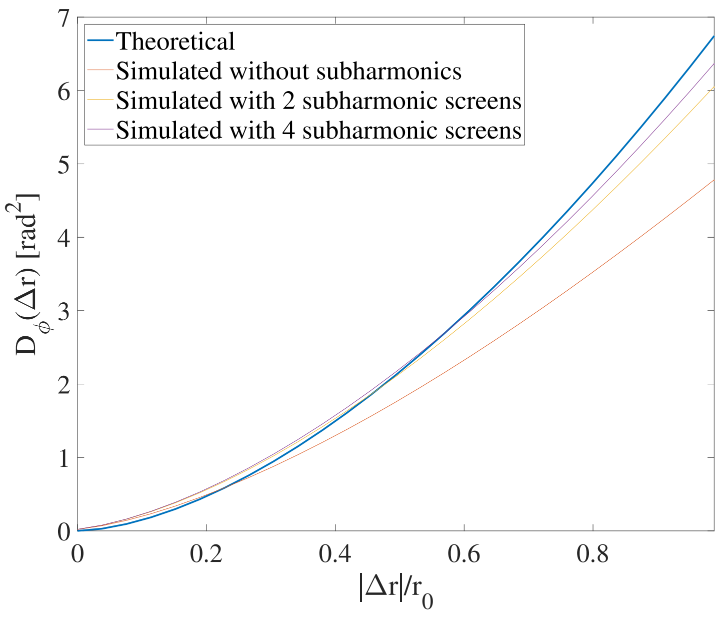


Figure 4.17: Comparison of theoretical and simulated structure functions. The addition of subharmonics greatly increase the accuracy of the synthesised phase screens. Values were obtained by averaging over 1000 screen realisations. As in the final simulator design, these screens were generated at larger scale $\tilde{N}$, and cropped down to N

functions obtained from the generated phase screens can be seen in Figure 4.17, where three phase screen structure functions are shown: two generated with the use of subharmonics and one generated without subharmonics.

This graph firstly shows that the generated screens are very similar to that of the theoretical structure function, indicating that the generated screens have statistically valid phase values. Secondly, it can be seen that the use of subharmonics provides significant improvement over the initial simulation method, especially at larger values of $|\Delta r|/r_0$. This is due to the fact that these values correspond to lower spatial frequencies, which lie at the centre of the modified von Kármán PSD. As presented in Section 4.3.2, the default frequency sample rate is unable to appropriately access these frequencies. These results therefore show that, by allowing access the lower frequencies of the PSD, the subharmonic method of phase screen generation is more accurate when compared to theory. It can also be seen that using more subharmonic screens allows the PSD to be sampled at even lower frequencies, resulting in a more accurate structure function at larger values of $|\Delta r|/r_0$. During simulation, the structure function for each individual phase screen along the propagation path can also be analysed, as shown in Figure 4.18.

Unlike Figure 4.17, this is simply a 'one-shot' view of the structure functions, as opposed to an average over multiple screens. Due to the random nature of

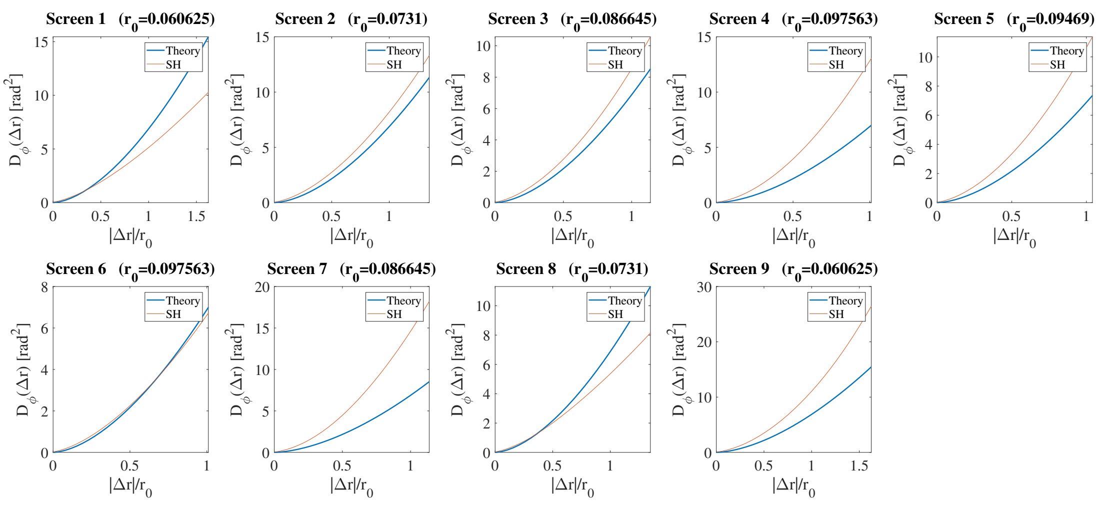


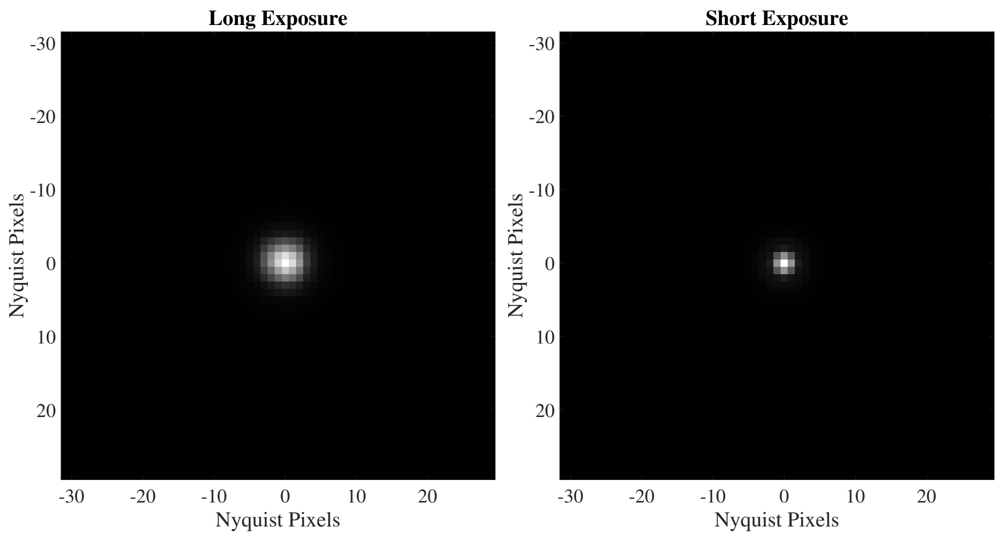
Figure 4.18: Structure function comparison for each screen generated during a single simulation.

the screen generation, this leads to some structure functions being more accurate than others when compared to the theoretical value. However, across all screens, the calculated structure functions are well within the order of magnitude that is expected. These graphs also show that the phase screens are being accurately generated for multiple different values of r_0.

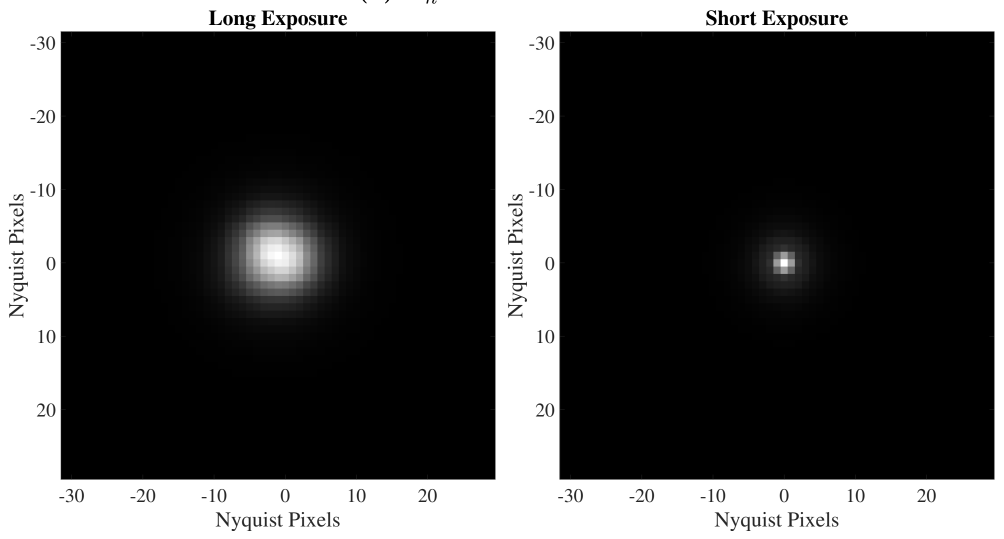
4.6.3 Point Spread Functions

With the validation of the propagation and phase screens complete, the remaining validation step is the validiation of the final image generation process. This can be done by comparing the generated PSFs to the theoretical PSF for an atmospheric model, as described in Chapter 2. In simulation, the LE PSF can be found by simply averaging over a large number of PSF realisations, which represents a long exposure version of the imaging system. To find the SE PSF, each PSF is first tilt-corrected, i.e. adjusted such that the peak of the PSF is moved to the centre coordinates. The same averaging procedure can then be performed over many realisations. This process was performed for two levels of turbulence, i.e. $C_n^2 = 0.25e^{-15}m^{-2/3}$ and $C_n^2 = 1.5e^{-15}m^{-2/3}$, and the resulting PSFs can be seen in Figure 4.19, as well as a comparison with the theoretical PSFs in Figures 4.20 and 4.21.

These figures firstly show that the size of both LE and SE PSFs increase with C_n^2, as expected, demonstrating that the increased phase values in the phase screens are indeed causing larger variations in the wave propagation. Secondly, it can be seen that the simulated PSFs correlate extremely closely to those of



(a) $C_n^2 = 0.25e^{-15}m^{-2/3}$



(b) $C_n^2 = 1e^{-15}m^{-2/3}$

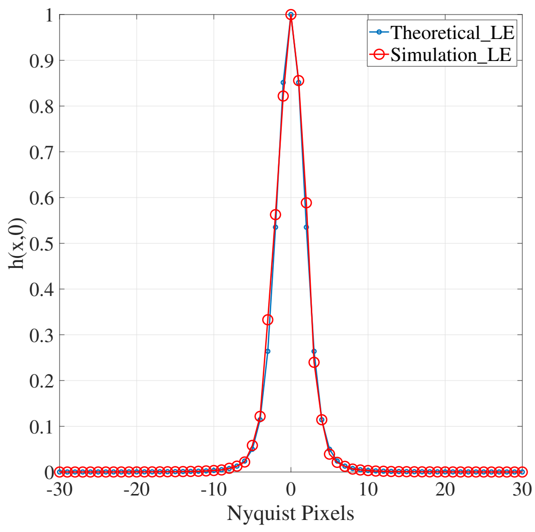
Figure 4.19: Long and short exposure PSFs for two different levels of C_n^2.

the theoretical PSFS, showing that the resulting size of the simulated PSFs are consistent with their expected theoretical counterparts.

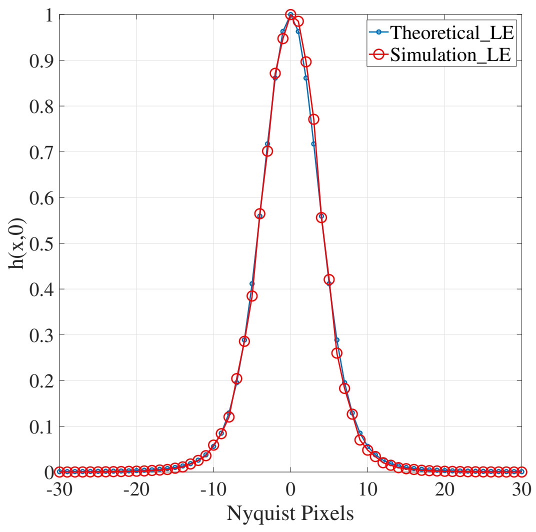
The validation steps presented allow the accuracy of the simulator to be clearly understood. It can therefore be confidently used to synthesise turbulent data.

4.6.4 Visual comparison with other simulation tools

A final validation step that can be undertaken is the comparison with other publicly available turbulence simulation tools. As mentioned previously, such tools were unavailable at the beginning of research, however since then, steps have been taken to close this gap in simulator availability. One such simulation

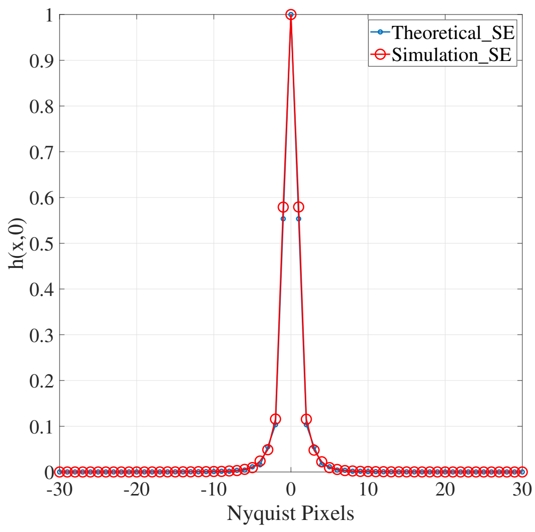


(a) $C_n^2 = 0.25e^{-15}m^{-2/3}$

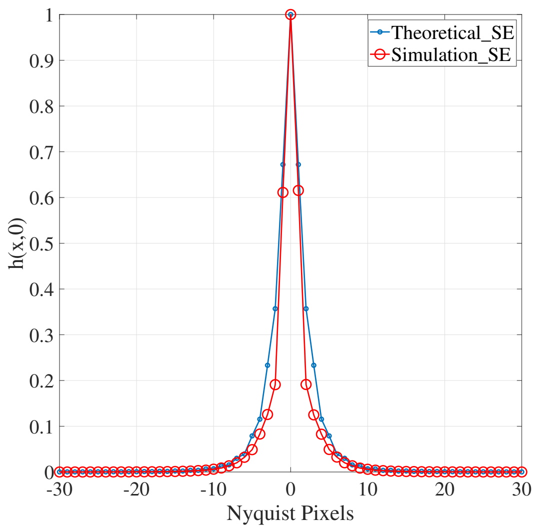


(b) $C_n^2 = 1.5e^{-15}m^{-2/3}$

Figure 4.20: Central slices of the Long Exposure PSFs from two different levels of C_n^2, compared against the theoretical LE PSF.



(a) $C_n^2 = 0.25e^{-15}m^{-2/3}$



(b) $C_n^2 = 1.5e^{-15}m^{-2/3}$

Figure 4.21: Central slices of the Short Exposure PSFs (Tilt-corrected) from two different levels of C_n^2, compared against the theoretical SE PSF.

tool is that from Mao et al. [76], who have made their simulator publicly available. This tool has been used in the UG^2+ competition track for turbulence mitigation.

By using the same settings in the simulator from Mao et al. and the simulator developed within this chapter, a visual comparison between the syntehsised images can be performed. Such images are shown in Figures 4.22 and 4.23 (With more examples provided in Appendix B).

From these example images, there are a number of key differences that can be observed. Firstly, for low levels of turbulence (Figure 4.22) the outputs from

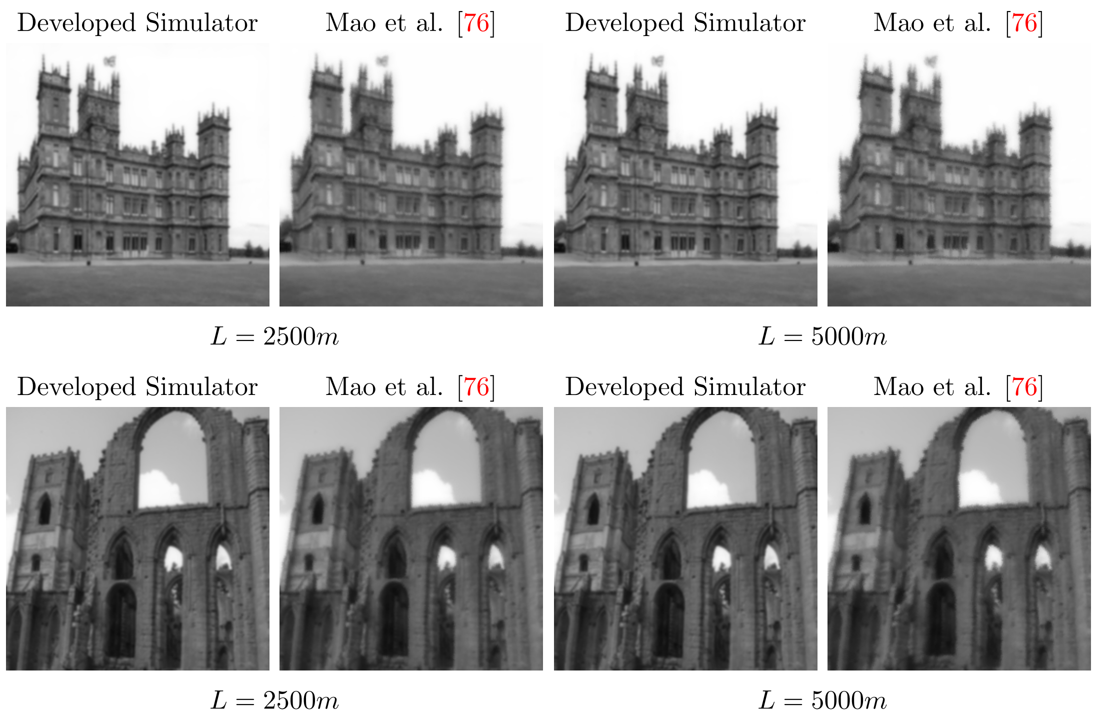


Figure 4.22: Example outputs from the developed simulation tool, compared with the work from Mao et al [76]. Two outputs from each are shown: $L = 2500m$ and $L = 5000m$. Both synthesised with a turbulence strength of $C_n^2 = 0.1 \times 10^{-15} m^{-2/3}$

the two simulators are similar for both values of L. This shows that both models capture the fact that, in calm atmospheric conditions, additional distance does not significantly increase distortion. However, as the strength of the turbulence increases, any additional distance between camera and object makes a larger impact on the image quality. This can be seen in Figure 4.23 where, again, both simulators are able to accurately capture this relationship.

It can be seen however that, for all settings shown, the output from the developed simulator is less distorted than that from the comparison simulator, especially in the case of higher turbulence. However, it is argued that the output from the comparison simulator is far too distorted for such settings. For example, the settings used for Figure 4.23 ($L = 5000$), correspond to a Fried parameter of $r_0 = 0.0458$. Given that the simulation was performed with an aperture diameter of $D = 0.1$, the resulting ratio of these two parameters is $\frac{D}{r_0} = 2.182$. As detailed in Chapter 2, such a ratio lies just above the threshold for what is considered weak turbulence. Given this, the images produced by the Mao et al. simulator visually seem too distorted. This is further supported by the comparison made in Figure 4.24, from which it can be seen that the Mao et al. simulator produces output characteristics that align closely with those of the real high turbulence example image. The outputs from the developed simulator however align with

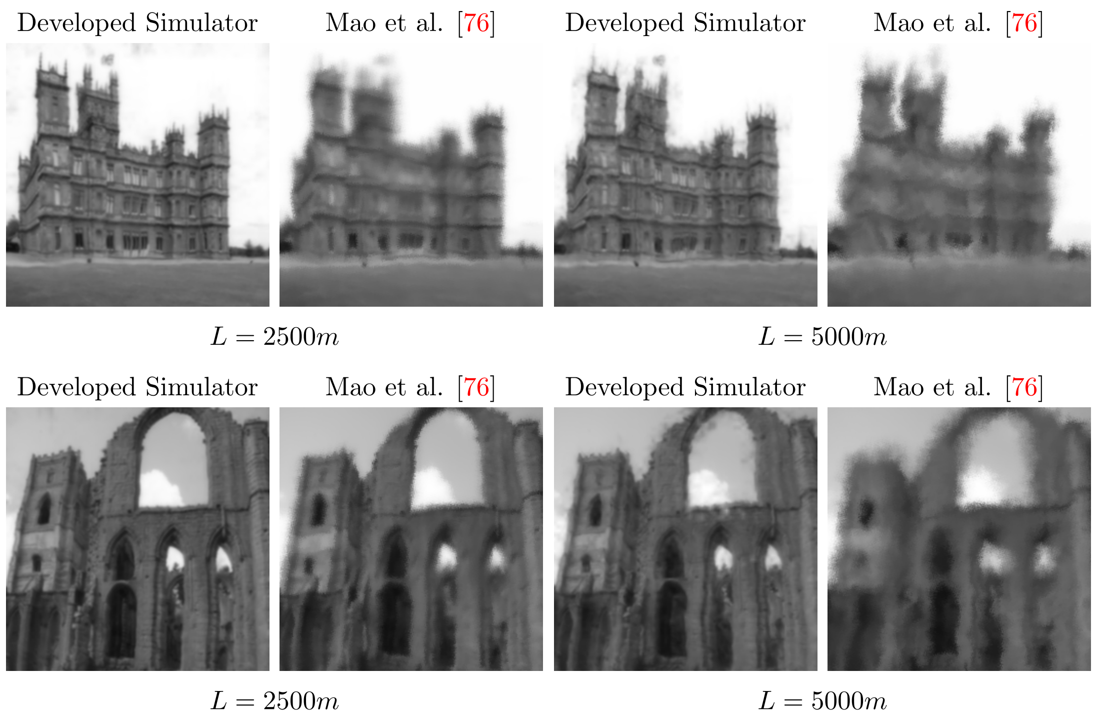


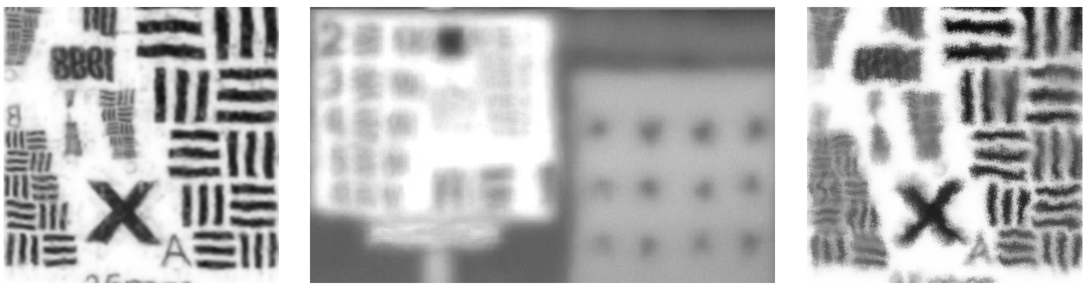
Figure 4.23: Example outputs from the developed simulation tool, compared with the work from Mao et al [76]. Two outputs from each are shown: $L = 2500m$ and $L = 5000m$. Both synthesised with a turbulence strength of $C_n^2 = 1.5 \times 10^{-15} m^{-2/3}$

the expectation of lower turbulence.

4.7 Dataset Synthesis

The remaining chapters of this thesis make use of Deep Learning algorithms, which require sufficient datasets in order to train. As highlighted in Section 4.1, the simulator described in this chapter was developed in order to facilitate such data generation. To this end, three turbulent datasets were synthesised, using two different source datasets.

To provide the Deep Learning algorithms with appropriate data, the image contents first had to be suitable for the task at hand. This required a source dataset consisting of scenes that could be legitimately corrupted by a turbulent atmosphere. Given this, the Places dataset [28] was chosen, which contains a total of 1,469,737 images within 205 different categories. Of these, 31 were chosen as categories that could be prone to turbulent interference, where categories were excluded if the contents were unlikely to be affected in a real scenario, such as indoor scenes or scenes at close ranges. The remaining categories contained images of outside scenes at varying ranges and environments. By isolating the



Developed Simulator ($\frac{D}{r_0} = 2.182$) Real turbulence [51] ($\frac{D}{r_0} = 7.1$) Mao et al. [76] ($\frac{D}{r_0} = 2.182$)

Figure 4.24: Simulator outputs compared against a real image captured during high turbulence [51].

data to these categories, a total of 148,884 images were selected as potential candidates for simulation.

The second source dataset was that of ImageNet [230], which contains 1,281,167 training images, 50,000 validation images and 100,000 test images. The motivation for choosing this dataset was the ability to perform classification with a pre-trained model. Therefore the effectiveness of any turbulence mitigation algorithm can be quantified by classification accuracy after processing. For turbulent dataset synthesis, the 50,000 images from the validation set were used, and were pre-processed such that they were all the same size (257x257). Any images smaller than this were simply ignored, and larger images had a central crop taken. This resulted in 47,169 total images for simulation.

Although the two source datasets provided RGB colour images, all images were first converted into greyscale in order for their use in the turbulence simulator.

During development, two different iterations of the simulator were utilised. The version described in this chapter is that of the final one, that allows the simulation of videos as well as single images. An earlier version was incapable of video simulation, and did not make use of phase screen subharmonics. The first dataset, ‘turbulence images’, was synthesised using this earlier version, with the remaining two datasets being synthesised with the final version.

Given that the Places and ImageNet datasets contain only single images, the following datasets are inherently static (i.e. the only movement is due to turbulence distortion). The motivation for this was the fact that turbulence mitigation is already a challenging task, and that initial attempts would be simplified if the difficulty was constrained to the turbulence strength itself, and not the video contents. Another factor that led to this decision is the fact that the simulations are run with a propagation distance of 5km. Under such long-range imaging

Table 4.2: C_n^2 ranges used for the four sub-datasets, all of which constitute the 'turbulence images' dataset. The number of images synthesised per sub-dataset is also provided. The C_n^2 values used during simulation were five evenly spaced values between the minimum and maximum of each range, inclusive

Sub-Dataset	**Minimum** C_n^2	**Maximum** C_n^2	Image Count
Low	$0.001 \times 10^{-13}\, m^{-2/3}$	$0.334 \times 10^{-13}\, m^{-2/3}$	3,100
Medium	$0.334 \times 10^{-13}\, m^{-2/3}$	$0.667 \times 10^{-13}\, m^{-2/3}$	3,100
High	$0.667 \times 10^{-13}\, m^{-2/3}$	$1 \times 10^{-13}\, m^{-2/3}$	3,100
Mixed	$0.001 \times 10^{-13}\, m^{-2/3}$	$1 \times 10^{-13}\, m^{-2/3}$	3,999

conditions, motion within the scene is expected to be minimal, therefore making the use of static videos a reasonable approximation.

4.7.1 Single Frame Dataset

The first dataset that was synthesised was the 'turbulence images', which made use of the first version of the turbulence simulator, and could therefore only synthesise individual images. This dataset consisted of four sub-datasets, representing 'Low', 'Medium', 'High' and 'Mixed' turbulence. The motivation for these sub-datasets was the desire to test new turbulence mitigation algorithms on increasing levels of difficulty. In order to ensure that each dataset was suitably distributed, for each dataset, 100 images were randomly selected from each of the 31 suitable categories, resulting in 3,100 images per dataset. For the 'Mixed' Dataset, this was increased to 129 random images, providing 3,999 total images. The corresponding C_n^2 values and number of images synthesised for each sub-dataset are shown in Table 4.2.

The remaining simulation parameters used for this dataset are shown in Table 4.3, where it is worth noting the absence of values for 'Sub_Count' and '$N_{subharm}$'. This is due to the fact that, at the time of synthesis, the use of subharmonics was not yet implemented.

The values shown in Table 4.3 were chosen in line with those from Hardie et al. [81], where the large value of L_0 aims to emulate turbulence according to Kolmogorov statistics. The wavelength λ was chosen such that it represents a mid-point of the visible spectrum. Finally the values of D and L were chosen to emulate a realistic telescope setup for long distance imagery, with the choice of D also factoring in to the resulting sample count N during simulation. This choice of D allowed this sample count to be 64, which resulted in a reduction in computation complexity.

Table 4.3: Simulation parameters chosen for 'turbulence images' dataset, alongside the varied C_n^2 values in Table 4.2. Note the parameters that were not yet implemented when this dataset was synthesised.

Parameter	Value
l_0	0.01m
L_0	300m
λ	525nm
Sub_Count	N/A
$N_{subharm}$	N/A
D	0.1m
L	5km
$nscr$	5 (4 non-zero)
$skip$	4
Image_Pixels	257
frame_count	1
turbSpeed	N/A
turbDirection	N/A

4.7.2 Video Datasets

Using the two source datasets, Places and ImageNet, turbulent video datasets were synthesized from each: 'turbulence videos' from Places and 'turbulent imagenet' from ImageNet. The settings used for simulation were the same for both datasets, and are shown in Table 4.4. The variables used for the simulations were 'C_n^2', 'turbSpeed' and 'turbDirection', the values of which are shown in Table 4.5, where for each video, the values were randomly chosen assigned. Given the potential values for each setting, a total of 160 different simulations were possible, therefore ensuring comprehensive, diverse datasets. The range of values for the refractive-index structure parameter C_n^2 was chosen such that a range of low to high turbulence was represented, the two extremes of which are shown in Figure 4.14.

As in the case of the single frame dataset, these settings were chosen in line with that of Hardie et al. [81], where the key change is in phase screen count. This was done to overcome the constraints applied during the Fried parameter calculation (Section 4.3.1). The frame count of 15 was selected to approximate the total depth provided by five RGB frames.

The 'turbulence videos' dataset consists of all 148,884 images selected from the Places dataset, and is split into 98,263 training videos, 5,956 validation videos and 44,665 test videos. This dataset was made publicly available [29] in order to

Table 4.4: Simulation parameters chosen for 'turbulence videos' and 'turbulent imagenet' datasets.

Parameter	**Value**
l_0	0.01m
L_0	300m
λ	525nm
Sub_Count	2
$N_{subharm}$	4
D	0.1m
L	5km
$nscr$	8 (7 non-zero)
$skip$	4
Image_Pixels	257
frame_count	15

Table 4.5: Ranges of the varied parameters for simulation of 'turbulence videos' and 'turbulent imagenet' datasets. The proposed values allow for 160 different possible combinations.

Variable	Values
C_n^2	$(0.25, 0.6525, 0.875, 1.1875, 1.5) \times 10^{-15}\,\text{m}^{-2/3}$
turbSpeed	$1, 2, 3, 4$
turbDirection	→↘↓↙←↖↑↗

facilitate further research using accurate turbulent data.

For 'turbulent imagenet', all 47,169 images were passed through the simulator. However, due to the cropping operation performed on the larger images, as well as the conversion to grayscale, the image subject may no longer have been recognisable for classification. Due to this, the dataset was reduced such that the clean, turbulence free, versions of the remaining images were 100% classifiable by the EfficientNetV2-L classification model [21]. This resulted in a dataset of 34,467 videos, which was not split train/validation/test as its purpose was simply as a test dataset.

4.8 Discussion

The simulation technique described in this chapter constitutes a split-step propagation technique that can accurately propagate a point source through a simulated atmosphere and generate accurate point spread functions. The versatility of such a simulation is extremely important, as not only can the base input parameters be changed for the users specific needs, but a varied C_n^2 profile can also be provided, as well as control over the speed, direction and number of frames in a video sequence. The largest downside to this type of simulation is that of computation time. Due to the fact that for each frame, each pixel requires a propagation, and the fact that a video sequence requires multiple frames, the time taken to synthesise a single video can be large. Depending on the hardware available, a 15 frame video can take anywhere from 12 seconds (NVIDIA A100) to 50 seconds (Nvidia GTX 1070) to synthesise. Therefore, if requiring a large dataset, the time required is a clear limitation. Furthermore, the simulation time scales poorly with N, where a value of 64 produces a single image in approximately 4 seconds, a value of 128 takes approx. 13 seconds and 256 takes approx. 46 seconds (for 10 propagation steps). Therefore, if the input settings result in a large value of N, the time required for dataset creation may be infeasible. Such times could be further exaggerated if more screens were required, or a smaller skip value was used. It is for this reason that the 'turbulence videos' dataset was made publicly available. By sharing this dataset, it allows future research to skip the complicated step of turbulence simulation, and focus on the mitigation algorithms directly. As well as the dataset, this chapter has provided a detailed description of how such a simulation process is performed, and will certainly assist in its understanding.

4.9 Conclusion

This chapter has presented the implementation details of a turbulence simulation tool. This simulator represents a turbulent atmosphere by first dividing it into discrete layers, where the atmosphere within each layer is characterised by a single phase screen. In order to simulate the effects of these phases screens on a light wave, a point source is propagated from a source plane through each phase screen in turn, until the observation plane is reached. At this point, the lens of a camera is simulated, and a point spread function is generated. This PSF represents how the turbulent atmosphere has diverted and affected the point source as it has travelled. This process is repeated for each pixel within the source image, where each pixel has a unique path through the atmosphere, as defined by its spatial location in 3D space. Once a propagation has been performed for each pixel, the resulting array of PSFs can be applied to the source image to synthesise a final turbulent image. During simulation, the propagation of nearby pixels would result in very similar optical paths; this allows a skip parameter to be provided, which allows the simulator to perform propagations over a sparser grid of pixels, therefore reducing the number of computations required. As well as single images, the simulator is also able to simulate video sequences by generating extended phase screens where, in between each frame simulation, these screens are laterally translated in a chosen direction by a chosen number of pixels. The next frame is then synthesised as normal. This translation of the phase screens results in a turbulent video that is temporally correlated. The simulator is successfully validated by the confirmation of the operation of each sub-process in turn, where the key validation technique is that of comparing the simulated elements with their theoretical counterparts. Using this method, the generated phase screens are found to be correctly generated, as well as the final PSFs of the system as a whole. Finally, the simulator was used to generate 3 unique datasets, each of which are used in the following chapters of this thesis. Of these datasets, the 'turbulence videos' dataset was made publicly available, therefore facilitating future turbulence mitigation research without the need for complex and time consuming simulation.

Chapter 5

Application of Deep Learning to Turbulence Mitigation

5.1 Introduction

As presented in Chapter 2, the removal of turbulence induced distortion in images and videos is highly desirable, as it facilitates post-processing steps such as tracking or identification. It recent years, it has been shown that deep learning significantly outperforms traditional image processing techniques in many different image processing applications. Examples of such fields are object classification [21], image segmentation [22, 121], pose estimation [231] or super resolution [7, 110, 232]. Such data driven approaches allow the models to learn the underlying behaviour of the task at hand, and adapt to unseen samples. This success therefore motivated a similar application for the complex task of turbulence mitigation. The distortion induced by a turbulent atmosphere can render objects completely unrecognisable, and traditional techniques can struggle to produce a clean image in such scenarios. Initial adoption of deep learning to imaging through turbulence provided extremely promising results, and were able to outperform traditional, image processing based, methods. [25, 33, 59, 66, 87, 142–144]

As discussed in Chapter 4, the acquisition of real turbulence data is a near impossible task, as it is extremely difficult to obtain a turbulence free image without control of the atmospheric conditions. Lack of data is a prevalent issue in many deep learning applications, as not only is suitably varied data required, a substantial amount of such data is also required. As reported in Chapter 3, existing deep learning approaches for turbulence mitigation overcome this issue with the use of simulation techniques. Similarly, the studies in this chapter make use of the datasets synthesised as described in Chapter 4. Other datasets that were utilised alongside these simulated datasets were the ATSyn-static [88] dataset and the Turbulence Text Dataset [60]. Note that all datasets used throughout this chapter contain 'static' videos only (i.e. the only movement in each sequence is due to turbulence distortion). As discussed in Chapter 4, the justification for this was to simplify the problem by restricting the problem to turbulence distortion only, and not the contents of the videos.

This chapter presents an analysis into the application of deep learning to turbulence mitigation. This is done as a series of three studies, with the first being a preliminary study into single image turbulence mitigation using off-the-shelf deep learning architectures. The second study expands this to turbulence mitigation for video sequences, and aims to optimise a single network architecture. The final study then aims to demonstrate state-of-the-art performance in video turbulence mitigation, by leveraging the insights from the previous two study's.

Given that previous works had already provided evidence that deep learning yields improved results when compared to traditional techniques [25, 33, 66], the goal of the first study was to find the optimal deep learning architecture that would best lend itself to turbulence mitigation. When this research was begun, the application of deep learning to turbulence mitigation was not widely reported. Whilst these techniques had been proven to outperform traditional methods, there was still a great deal to learn regarding the optimal architecture design. The motivation of the first study was therefore to evaluate which network architecture would be able to adapt best to different data, without any architectural changes. The outcomes of such a study would then be extremely beneficial for any future development of turbulent specific deep learning models. The networks under review were originally designed for tasks such as image denoising, super resolution or image deblurring. In line with the nature of the models used, and the common practice of image deblurring/super resolution models at the time [110], only single frames were used as inputs to the models, therefore making use of the 'turbulence images' dataset synthesised in Chapter 4.

Traditional turbulence removal techniques, however, commonly make use of video sequences to improve the quality of their outputs. The added temporal information provided by video sequences allow a better understanding of the underlying, turbulence free, scene. Therefore, the second study of this chapter was the investigation into applying deep learning to turbulent videos. As with the first study, when this research was conducted, there was a paucity of deep learning turbulence mitigation models. It was therefore beneficial to make use of an existing deep learning model that was able to process video sequences. This study therefore made use of the Enhanced Deformable Video Restoration (EDVR) architecture [233], that was designed to perform various video restoration tasks, and was the winner of the NTIRE 2019 challenge for image deblurring. Using the original EDVR as an initial baseline, this study carried out two investigations into both the architecture itself and the loss function. Firstly, the benefit of the deformable convolution layers, used for image alignment, was examined. Secondly the performance of the model when trained with a perceptual loss alongside the original Charbonnier loss was also examined. The aim of these investigations was to best determine how best to train a model when dealing with turbulent data, and to identify potential avenues of improvement in the specific case of turbulence mitigation. In order to facilitate this study, the datasets 'turbulence videos' and 'turbulent imagenet' were utilised.

The final study of this chapter aimed to use the key learning points from the previous two studies, and apply them to a state-of-the-art turbulence mitigation model. The model chosen was the Deep Atmospheric TUrbulence Mitigation (DATUM) [88] model which, as well as providing state-of-the-art performance, was also open source. From the results of the first two studies, two key findings were highlighted. Firstly, the use of video sequences greatly increased the chance of successful turbulence mitigation, and secondly, the loss function can improve the performance of a model, without any significant architecture changes. To apply such learnings to the DATUM model, its base architecture was left unchanged, as it was already capable of processing video sequences. Its loss function however was investigated for potential improvement. This investigation included the addition of a perceptual loss, as well as the removal of the tilt loss found in the original loss function within DATUM. As with EDVR, it was believed that the addition of a perceptual loss would allow DATUM to generate more perceptually pleasing images, therefore resulting in improved classification results. The removal of the tilt loss was motivated by its requirement of specialised data, which may not always be available. If DATUM was able to be trained without this loss, its use cases would expand significantly. Finally, a comparison between the EDVR and DATUM models was performed. Where, as well as turbulence mitigation performance, their implementation details were also compared. This allowed any advantage of each model to be highlighted.

The novelty of this chapter lies in its thorough investigation of how deep learning can be optmised for atmospheric turbulence mitigation. Unlike prior work, which primarily aimed to demonstrate that deep learning can outperform traditional restoration techniques, this chapter contributes a structured analysis of which architectures and loss functions are best suited to the unique challenges posed by turbulence distortion, and why. In particular, this chapter demonstrates that a perceptual-loss offers substantial benefits to the task of turbulence mitigation, without requiring specialised architectural modifications. It is shown that, for this task, the perceptual quality of the output images should be prioritised, as higher perceptual quality improves downstream performance. Consequently, incorporating perceptual losses enables models to produce higher-quality outputs. This principle is applied to a state-of-the-art model, where it is found that an improvement can be made via los-function optimisation.

The remainder of the chapter is organised as follows. Section 5.2 first provides the implementation details of the off-the-shelf single frame deep learning architectures that were retrained on turbulent data, with Section 5.3 then providing details on the EDVR architecture and the various studies performed with its design and training process. Section 5.4 then details the DATUM architecture and the different variations of the loss function that were tested. This section also shows how the DATUM and EDVR architectures were compared. Section 5.5 then presents the results to each of these sections in turn with Section 5.6 providing a detailed discussion. Section 5.7 finally concludes this chapter.

5.2 Single Frame Turbulence Mitigation

To understand the optimal deep learning architecture design for turbulence mitigation, an initial approach was the utilisation of off-the-shelf Deep Learning (DL) architectures that were originally designed for image processing tasks such as image deblurring or super-resolution. The motivation for this was firstly due to the lack of existing DL turbulence mitigation models at the time of study, and secondly, to identify if these networks would be able to adapt to turbulent data without significant changes to their architectures. For this work, the 'turbulence images' dataset described in Chapter 4 was used. Where the 'Mixed' sub-dataset was used for training, and the 'Low', 'Medium' and 'High' sub-datasets were used for testing. The architectures under review were: BRDNet [234], RDN [235], SuperSR [236], CAE-Unet-CAE [33], RCAN [237] and DnCNN [136]. Of these networks, CAE-Unet-CAE was the only architecture originally designed for turbulence mitigation and the DnCNN architecture has been similarly applied to turbulence mitigation in other works [56, 66]. The following section provides a high level description of each model, the full architectures of which can be seen in Appendix A.

5.2.1 Comparison Models

BRDNet: Originally designed for denoising, the architecture from Tian et al. [234] makes use of batch renormalisation and dilated convolutions. Their overall network is made up of two parallel paths (one with dilated convolutions, the other with normal convolutions), the outputs of which are concatenated to provide a predicted noise map that can be subsequently subtracted from the original input image.

RDN: The Residual Dense Network from Zhang et al. [235] makes use of Residual Dense Blocks (RDB) within their network. These blocks employ local residual learning and densely connected convolutional layers followed by a 1×1 kernel feature reduction. These blocks are stacked, such that the output of each is concatenated together to act as a global feature fusion operation.

CAE-Unet-CAE: The architecture from Chen et al. [33] is the only architecture in this work that was initially designed to perform turbulence mitigation. It consists of three networks: CAE_1, U-Net and CAE_2. The first CAE (Convolutional Autoencoder) extracts features that are then fed into the U-net structure for noise removal. The final CAE then reconstructs a clean image. In order to train their network, the authors use the theoretical PSF for atmospheric turbulence to generate synthetic data.

RCAN: The architecture from Zhang et al. [237] introduce the Residual Channel Attention Block (RCAB), and was designed for the task of super resolution. The architecture consists of a Residual In Residual (RIR) structure, where a series of Residual Groups are placed in series and within each are RCABs, also placed in series. In the original architecture, the output of the RIR structure is upscaled with the use of ESPCNN [238]. This was omitted for this investigation, as the goal was the removal of turbulence from an image, without changing the spatial dimensions.

SuperSR: The model proposed by Feng et al. [236] was initially designed for the task of super resolution and was runner up in the 2019 NTIRE super resolution challenge [110]. Their network consists of a U-net structure, where the 'main network' processes the features at the lowest resolution. Within this, several 'Cascading blocks' are connected together in a densely connected fashion. Each of these cascading blocks contain a similarly densely connected structure of RCABs (as introduced by Zhang et al. [237]).

DnCNN: The DnCNN architecture from Zhang et al. [136] was initially designed for the task of denoising, and was commonly used as a comparison for newer networks [107].

The input to the architecture is passed through a convolutional layer with ReLU (Conv+Relu) activation, followed by 17 'Conv+BN+ReLU' layers. The final residual image is a prediction of the noise present in the input, which can then be subtracted from the original to provide a clean output image.

This model had been adapted for the task of turbulence mitigation before [56, 66]. In the case of [56], the model was used as a component within a larger turbulence mitigation framework, where it was used to replace more traditional blur estimation and unsharp masking operations. Gao et al. [66] adapted the model to perform full, end to end, turbulence mitigation, and improved performance by increasing the kernel size of the network to 5×5, which allowed for a greater receptive field to better capture the full range of turbulent distortion.

5.2.2 Training Procedure

The networks were trained with the details provided in the original works. For each, the base architecture structure remained unaltered with the exception of RDN and RCAN due to memory limitations. In the case of RDN, the feature size of each layer was reduced from 64 to 32. For RCAN the number of Residual Groups in the model was reduced to 5 (from 10), with each containing only 10 RCABs (reduced from 20). The DnCNN model was implemented as inspired by Gao et al.[66], with an increased kernel size of 5×5 and a network depth of 17 layers.

The loss function for each network was kept as intended in the source papers and the ADAM optimiser was used for each, with $B_1 = 0.9$, $B_2 = 0.999$ and $\delta = 10e^{-8}$.

In order to identify the optimal learning rate and batch size for the training process, a hyperparameter search was performed for each model. Furthermore, the number of training epochs was tailored for each network, as different networks would train faster than others. The networks given 10 epochs of training had their learning rate halved after the 5th and 8th epochs. Whereas, the networks given 15 epochs of training, had their learning rate halved after the 9th and 12th epochs. The final training parameters used for each model can be seen in Table 5.1.

Each model was trained using the single frame 'Mixed' turbulence sub-dataset, consisting of 3,999 turbulent images with a wide range of turbulent distortion. For the purpose of training, 79,900 patches of size 80×80 were extracted from the available images. This was then further split into training and validation

Table 5.1: Training hyperparameters used to train each DL model for the task turbulence mitigation.

Network	Learning Rate	Loss	Optimiser	Batch Size	Epoch Count
BRDNet [234]	0.00001	L2	ADAM	16	10
RDN [235]	0.0001	L1	ADAM	10	15
CAE-Unet-CAE [33]	0.001	L2	ADAM	16	15
RCAN [237]	0.0001	L1	ADAM	10	15
SuperSR [236]	0.00001	L1	ADAM	16	10
DnCNN [136]	0.00001	L2	ADAM	10	10

L1: Least Absolute Error **L2**: Least Square Error

sets with a ratio of 0.8, resulting in 63,920 training patches and 15,980 validation patches. Once trained, the models were tested using 1,000 images from each of the ‘Low’, ‘Medium’ and ‘High’ sub-datasets (Randomly sampled from the available 3,100). This allowed the analysis of each models ability to generalise over different distortion intensity levels. The metrics used for reconstruction quality assessment were the Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM) [171].

5.3 Video Turbulence Mitigation with EDVR

The contextual information provided by a single image is very limited, especially if that image has been distorted. This greatly increases the difficulty of developing image recovery algorithms for single images. Traditional turbulence mitigation algorithms commonly make use of video sequences rather than single images [12]. This therefore motivates the use of videos in a deep learning setting as well. This section presents an analysis of an off-the-shelf deep learning model, the Enhanced Deformable convolution Video Restoration model (EDVR)[233], for the purpose of turbulence mitigation in videos.

The EDVR model was originally designed for two tasks: super resolution and deblurring. It was the winner of both competition tracks in the NTIRE 2019 video restoration and enhancement challenge [140], remaining unbeaten the following year. The success of this model in two domains (super resolution and deblurring) provided reassurance that it would also be able to perform well on turbulent data, thus its selection for this investigation.

The architecture is well suited for turbulence mitigation, as it follows the same principles found in traditional turbulence removal algorithms. The EDVR architecture first aims to align each of the frames relative to the central frame. It then utilises the added information provided by the video sequence to better

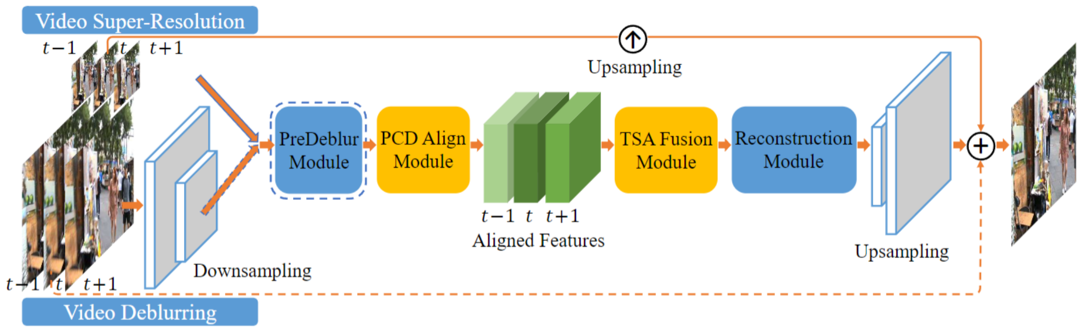


Figure 5.1: EDVR Architecture [233]. A framework designed for various video restoration tasks such as super resolution or deblurring. The input to the model is downsampled to reduce computation, and a single high quality image is generated from a single video sequence (Many to One).

understand the spatial and temporal distortion, and selectively chooses frames that best benefit the image recovery.

As can be seen in Figure 5.1, the architecture consists of four modules: PD (PreDeblur), PCD (Pyramid, Cascading, and Deformable) alignment, TSA (Temporal and Spatial Attention) fusion, and Reconstruction. When performing video deblurring, the input video is first downsampled by a factor of 4 before processing, and is upsampled again at the end of the model. This downsampling allows the computations to be performed in a low resolution space, therefore reducing computational complexity. The model has a 'many to one' design, whereby the input is a series of frames stacked together, and the output is a high resolution/deblurred representation of the central frame.

The Predeblur and Reconstruction modules consist of cascaded residual blocks, which are common building blocks found in image processing DL models [237, 239]. The purpose of the Predeblur module is to aid the PCD module by preprocessing the input frames, and therefore improving the quality of alignment. The Reconstruction module takes the output feature map from the TSA module and performs the necessary processing to complete the image recovery process. The feature map is upsampled to the original image size via pixel shuffle operations [238]. The output residual image is then combined with the centre frame of the original video sequence to generate a high-resolution output image.

Pyramid, Cascading and Deformable alignment Module

The purpose of the PCD alignment module is to align the contents of each frame with the centre reference frame. This is achieved via a cascading pyramid architecture, as shown in Figure 5.2. The input features are sequentially downsized

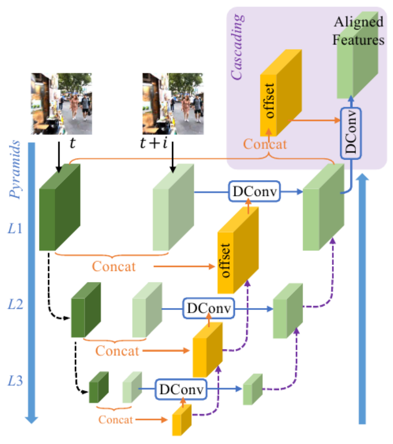


Figure 5.2: Pyramid, Cascading and Deformable convolution (PCD) Alignment Module [233].

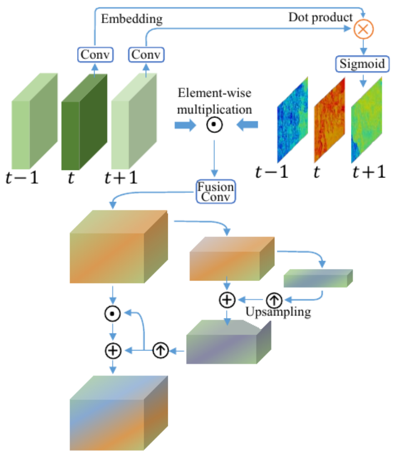


Figure 5.3: Temporal and Spatial Attention (TSA) Fusion Module [233].

to form 3 layers; this stepped approach allows the model to perform alignment in a course-to-fine manner. Within each layer, the current frame $(t+i)$ and the reference frame (t) are concatenated. Using this feature map, the model identifies the difference between the two frames and generates a set of offsets that represent this difference. These offsets, alongside the current frame $(t+i)$, are passed into a deformable convolution (dcn) layer [119], the output of which then represents the current frame aligned with the reference frame. This process is performed at each level, with the outputs of each level cascading back up the pyramid, until the final aligned feature map is obtained. The deformable convolution layer allows the kernel weights to access locations outside of its traditional $k \times k$ grid (where k is the kernel size). This aids in the task of alignment by enabling the kernels to 'seek out' features corresponding to the same spatial location from the reference frame. This approach allows them to move beyond the constraints of their local area.

Temporal and Spatial Attention fusion Module

The goal of the TSA module is that of image/frame fusion. Its purpose is to take a series of input frame features and combine them in such a way that the resulting feature map represents the best spatial features from each frame, as shown in Figure 5.3.

It achieves this by first exploiting the added temporal information of a video sequence, using temporal attention. For each frame, the similarity to the reference frame is calculated, and subsequently used to apply greater focus to the frames that are more similar to the reference.

Once each frame has been temporally weighted, the fusion into a single output feature map is achieved via a fusion convolution layer, the output of which is a fused feature map of all input frames. This feature map can then be further improved via spatial attention. To achieve this, the model creates spatial attention masks by downsizing the feature map from the temporal attention. Once downsized to the lowest resolution, the remaining spatial features represent the areas of the image that require the most attention. After being upsampled once more, the feature maps represent spatial attention masks, indicating key areas of the input feature map. The mask is formed of two parts, a multiplicative part and an additive part. The combination of these allows any spatial location within the feature map to be highlighted or reduced. The final output of the TSA is the multiplication and addition of the spatial attention masks, resulting in a feature map that represents a high quality output image.

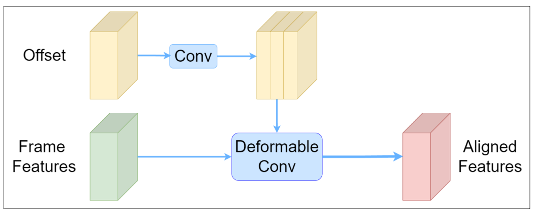
5.3.1 Modified EDVR Models

In order to train EDVR, the model first required an alteration in the input shape where, in the original work, five distorted RGB frames were used. However, the turbulence dataset used in this work consisted of grayscale images only. In order to compensate for this lack of RGB colour channels, the input was set to 15 grayscale frames, thereby maintaining the same total input size. This change may also lead to a further enhancement, as the model now gains the temporal knowledge of fifteen frames as opposed to five. As well as this change in input shape, another change that was made to facilitate training was the use of a smaller version of EDVR, named EDVR$_s$, as explained below. The motivation for this was a simple matter of hardware limitations at the time of training. This would not pose a significant problem, as this smaller model was used in all tests regarding the changing of the architecture, therefore the cross comparisons performed between such models were still valid.

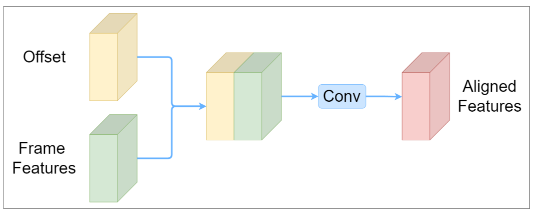
The following sections detail the various experimentations performed on the architecture and training procedure of EDVR.

Deformable Convolution

One notable design choice from the original EDVR architecture was that of use of deformable convolution layers, inspired by the TDAN model [240], within the PCD alignment module. It was found that initial attempts at training the EDVR model for turbulence mitigation were unsuccessful. As described in Section 5.3,



(a) Original deformable convolution



(b) Proposed replacement

Figure 5.4: Original deformable convolution [119] layout alongside the proposed replacement layout. In both cases, the input offset and frame features are combined to create a feature map aligned with that of the reference frame.

the deformable convolution generates offsets using learned weights, where these offsets allow the convolution kernel to look 'outside' its original spatial grid. During training however, these offsets would grow too large, resulting in the model being unable to train. To overcome this issue, the model training can be resumed from a previous state, where the offsets have not yet become unstable. This however is a frustrating task and may be required multiple times per training. Given this complexity in model training, it was proposed that the deformable convolution layer could be replaced with a much simpler CNN layout, as can be seen in Figure 5.4. This replacement simply takes the learned offsets from the PCD module and concatenates them with the current frame features, before being processed by a single convolution layer. For this investigation into the presence of DCN, two models were trained: the original model (EDVR_s), trained by resuming training, and a model with the replacement layout ($\mathrm{E\overline{D}VR}_s$). Note that EDVR_s is the only model in this section that makes use of the original DCN layers, and all other models make use of the replacement layout illustrated in Figure 5.4b.

As well as investigating the *replacement* of the DCN layer, another solution to the unstable training could be the process of restricting its learnable offsets. By setting a maximum value for the offsets with a clipping threshold, any unstable growth can be mitigated. However, this would cause the clipped offset weights to have zero gradients during backpropagation, potentially leading to training limitations. With this taken into consideration, four models were trained with four different threshold levels for weight clipping. These thresholds were: 1, 2, 5 and 10. By comparing their results to an unconstrained version, any advantages/disadvantages to the clipping procedure can be understood.

Perceptual Loss

The default loss function used to train EDVR is that of the Charbonnier loss function [233], which measures the squared difference between the output image and the corresponding ground truth. This therefore steers the model to produce images that are as close to the ground truth as possible. This does not however guarantee good perceptual quality. Turbulence mitigation is commonly used as part of a larger pipeline, where the final objective is more likely to be that of object identification, tracking or classification. It would therefore be desirable that the loss function, alongside optimising the model to recreate the ground truth, could also promote perceptual features, therefore aiding post-processing algorithms. To this end, incorporating a perceptual loss [241] alongside the traditional loss would be expected to improve turbulence mitigation models, enabling them to integrate more effectively into larger frameworks.

Perceptual loss is calculated with a pre-trained classification model, ϕ, which is used to encode each output image into its feature maps at each layer. The activations of the j^{th} layer can then be compared with the corresponding activations from the ground truth image. This is described as

$$\mathcal{L}_{percep}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \left\| \phi_j(\hat{y}) - \phi_j(y) \right\|_2^2 \tag{5.1}$$

where $\phi_j(\hat{y})$ are the activations from the j^{th} layer given the output image and $\phi_j(y)$ are the activations from the j^{th} layer given the target (ground truth) image. C_j, H_j and W_j are the number of channels, height and width of the j^{th} layer respectively.

Another loss function that would aim to steer turbulence mitigation models towards promoting perceptual features is that of detection loss. Unlike perceptual loss, which computes the L2 difference between intermediate feature maps, detection loss evaluates the output of a classification model. The limitation of such a loss function however is the reliance on labelled data. When using detection loss, it is the difference between the predicted label and the ground truth that provides feedback. Therefore, in order to train a turbulence mitigation model with a detection loss, the data would require accurate classification labels. The perceptual loss avoids this by simply using a pre-trained classification model to extract meaningful features. The loss then reflects how similar the model's output is to the ground truth in terms of perceptual quality, rather than binary classification accuracy.

For this work, the activations for the perceptual loss were extracted from the 'conv5_4' layer of the VGG19 [120] pretrained model.

Meanwhile the original loss function, the Charbonnier loss, is defined as

$$\mathcal{L}_{char}(\hat{y}, y) = \sqrt{\|\hat{y} - y\|^2 + \varepsilon^2} \tag{5.2}$$

where $\hat{y}$ is the output image, y is the target image and ε is a small constant to ensure differentiability at zero.

The final proposed loss, $\mathcal{L}$, can then be constructed from (5.2) and (5.1) as

$$\mathcal{L} = \lambda \mathcal{L}_{char} + \alpha \mathcal{L}_{percep} \tag{5.3}$$

where λ and α are weighting values that control the influence of each loss. To fully investigate the impact of perceptual loss, two models were trained, each with different ratios of λ and α, first a model where $\lambda/\alpha = 0.75/0.25$ ($\text{E}\overline{\text{D}}\text{VR}_{s25}$), and secondly a model where $\lambda/\alpha = 0.5/0.5$ ($\text{E}\overline{\text{D}}\text{VR}_{s50}$).

Model Size and Ablation Study

Due to limited access to suitable computational hardware at the beginning of this investigation, the size of the model was reduced. As previously mentioned, this smaller model was named EDVR_s. For this reduction, the number of features in each layer was reduced from 128 to 64 and the number of residual blocks in the reconstruction module was reduced from 40 to 10. Access to improved hardware at a later stage did enable the training of an original version of the model. Given the previously mentioned difficulties in training the original model with the deformable convolution layers, this larger model was trained using the replacement architecture as shown in Figure 5.4. This model is subsequently labelled $\text{E}\overline{\text{D}}\text{VR}$. This larger model therefore also allows an analysis of the impact of model size. Using this, one can decide whether it is worth compromising on performance for a smaller more deployable model. An ablation study of $\text{E}\overline{\text{D}}\text{VR}_s$ was also conducted, evaluating the presence of the PreDeblur (PD), PCD and TSA modules, allowing an understanding of which modules are contributing most to the overall performance. This also allows the original model design to be scrutinised when being trained with turbulent data, as this change in image content may result in one or more of the modules not performing as expected. The eight resultant trained models represented all combinations of these three modules, and a version that contained none of the modules; the chosen naming convention for these experiments is shown in Table 5.2.

Table 5.2: Assigned names and description for each version of EDVR trained for ablation study.

Model Name	Description
PD_PCD_TSA	All modules present
PD_PCD	No TSA Module
PD_TSA	No PCD Module
PCD_TSA	No PD Module
PD	Just the PD Module
PCD	Just the PCD Module
TSA	Just the TSA Module
None	None of the Modules

5.3.2 Training Procedure

As in the original work, models were trained using the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was initialised as 6×10^{-4} (2×10^{-4} for E$\overline{\text{D}}$VR), and was adjusted according to a ReduceLROnPlateau scheduler, with a patience of 20 epochs and a reduction factor of 0.5, where the adjustments were made according to the loss evaluated on the validation set of data. All models were trained from scratch until the learning rate dropped below a threshold value of 2×10^{-6}. The motivation for retraining a model from scratch, rather than using pre-trained weights, is that of flexibility. When performing transfer learning, where a pre-trained model is adapted to new data, the weights of the source (pre-trained) model must be compatible with the architecture of the target model. This therefore imposes constraints on the design and adaptation of the architecture. Due to the nature of the various configurations of EDVR that are proposed in this section, the constraint of this approach would result in unfair comparison. Therefore, to ensure that each model is trained for fair comparison, no pre-trained weights are utilised. A summary of the different configurations of EDVR can be seen in Table 5.3.

The training data used for these models was that of the 'turbulence videos' dataset developed in Chapter 4, consisting of 148,884 turbulent videos, each with 15 frames. The dataset covers a wide range of turbulence, as well as varied turbulence 'speed'. The dataset is separated into a train/validation/test split of 98,263/5,956/44,665 images respectively.

As described previously, the most common use of a turbulence mitigation algorithm would be that of pre-processing images for tasks such as tracking or classification, and that the addition of a perceptual loss may improve a models performance in this regard. Therefore, to test this hypothesis, a new method of

Table 5.3: Summary of the different configurations of EDVR trained for analysis.

Model Name	Deformable Convolution	Original Size	λ	α
EDVR_s	✓	✗	1	0
$\mathrm{E\overline{D}VR}_s$	✗	✗	1	0
$\mathrm{E\overline{D}VR}_{s25}$	✗	✗	0.75	0.25
$\mathrm{E\overline{D}VR}_{s50}$	✗	✗	0.5	0.5
$\mathrm{E\overline{D}VR}$	✗	✓	1	0

image quality assessment is proposed in the form of post-processing classification. If the classification accuracy on the outputs of $\mathrm{E\overline{D}VR}_{s25}/\mathrm{E\overline{D}VR}_{s50}$ exceeds that of the output images from $\mathrm{E\overline{D}VR}_s$, it would indicate that the inclusion of perceptual loss is indeed beneficial. To facilitate this new metric, the EfficientNetV2-L classification model [21] was used alongside the 'turbulent imagenet' dataset described in Chapter 4, which contains accurately labelled data. The output images from each model are passed to the classifier to assess if an improvement in accuracy is achieved post-mitigation. Before simulation, each ground truth image from 'turbulent imagenet' was confirmed as classifiable with 100% accuracy by EfficientNetV2-L. The motivation for this was to ensure that the only reason for an unclassifiable image would be that of turbulent distortion, and not due to the classifier. This dataset was used only for testing purposes.

Alongside the classification accuracy, more image quality metrics are employed to quantify the outputs of the models. As well as PSNR and SSIM, two more Full Refernce (FR) metrics are used: CW-SSIM [221] and RMSE as well as four No Reference (NR) metrics: BRISQUE [213], NIQE [216], CLIPIQA [214] and MANIQA [180]. The motivation for the use of these metrics is to better gain an understanding of image quality by referencing multiple metrics at once.

5.4 Video Turbulence Mitigation with DATUM

Over recent years, the field of turbulence mitigation with deep learning has become more and more popular, with new turbulence mitigation architectures constantly being developed. The DATUM architecture [88] is one such model, providing state-of-the-art results. For the design of DATUM, the authors took inspiration from traditional turbulence mitigation techniques, and aimed to follow similar pipelines, using modern deep learning. This is presented as three distinct steps: Feature Registration, Temporal Fusion and post-processing. These steps

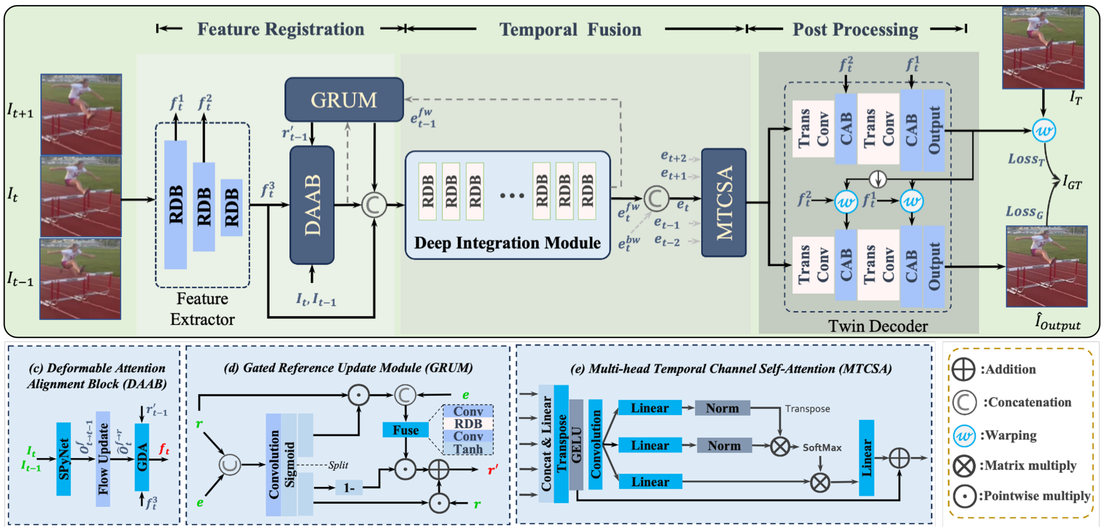


Figure 5.5: DATUM Architecture [88]. Showing the processing of the t-th frame in the input sequence.

can be seen, alongside the full architecture of DATUM in Figure 5.5.

The purpose of the feature registration stage is to first construct a reference frame from the provided input images. Each image is then aligned to this reference frame. Within DATUM, this is achieved by first extracting features from the input with a series of spatially decreasing Residual Dense Blocks (RDB) [235], extracting features at each level as $f_t^{\{1,2,3\}}$. Where $t \in \{0, 1, 2, \dots, N-1\}$, and N is the total number of input frames.

The final feature map, f_t^3, is then passed into the Deformable Attention Alignment Block (DAAB), which is also provided a downsized version of the current frame, I_t, a downsized version of the previous frame, I_{t-1}, and a reference map from the previous frame, r'_{t-1}. This reference map is generated recursively by the Gated Reference Update Module (GRUM).

The DAAB first uses the SpyNet [242] to estimate the flow, $O^f_{t\rightarrow t-1}$, between the two frames I_t and I_{t-1}, which is then used to update the estimated flow between the current frame and the reference, $\hat{O}_t^{f\rightarrow r}$. The final registered feature map, f_t, is then found using a Guided Deformable Attention (GDA) block. The DAAB is employed in a bidirectional manner, in that, for each feature map, f_t^3, a registered feature map, f_t, is evaluated for both forwards and backwards optical flow.

To then perform temporal fusion, the resulting registered features, f_t, are concatenated with the original feature map f_t^3 and the reference map r'_{t-1} to be passed into the Deep Integration Module (DIM) consisting of 15 RDB blocks. The output of which represents the forward and backward embeddings e_t^{fw} and e_t^{bw}.

After concatenation, these are then passed to the Multi-head Temporal-Channel Self-Attention (MTCSA) module for final frame fusion. Each are also fed back to the GRUM module to update the reference map for the subsequent frame, I_{t+1}.

Up to this section of the model, all input frames have been processed the same. For the MTCSA however, the input is defined by the chosen number future frames (ff) and past frames (pf) that should be used for the fusion step. The number of output frames from the model, N_{out}, is therefore limited to

$$N_{out} = N - ff - pf \tag{5.4}$$

and the input to the MTCSA is therefore,

$$(e_{t-pf}, \ldots, e_t, \ldots, e_{t+ff}), \quad t \in \{pf, pf+1, \ldots, N-ff\} \tag{5.5}$$

Finally, the output frames are generated by the post-processing stage, implemented with a twin decoder structure. For each frame, the feature map from the MTCSA is fed into two separate decoding paths. The upper path aims to undo any tilt present in the frame, whilst the lower path makes use of the upper paths tilt predictions to focus on the removal of blur. These paths are reinforced by the two loss functions used to train DATUM.

The first loss function, $\mathcal{L}_{char}$, is simply that of a Charbonnier loss evaluated between the output frame, $\hat{\boldsymbol{I}}$, and the ground truth image, $\boldsymbol{I}_{\mathrm{GT}}$. i.e.

$$\mathcal{L}_{char}\left(\boldsymbol{I}_{\mathrm{GT}}, \hat{\boldsymbol{I}}\right) \tag{5.6}$$

The second loss, $\mathcal{L}_{tilt}$, is the Charbonnier loss evaluated between the ground truth and the 'tilt corrected' image, denoted as $\mathcal{W}\left(\boldsymbol{I}_{\mathrm{tilt}}; \hat{\mathcal{T}}^{-1}\right)$, where $\boldsymbol{I}_{\mathrm{tilt}}$ is the 'tilt-only' image and $\hat{\mathcal{T}}^{-1}$ is the estimated inverse tilt field from the upper path of the twin decoder. The full tilt loss is therefore defined as,

$$\mathcal{L}_{\mathrm{tilt}} = \mathcal{L}_{\mathrm{char}}\left(\boldsymbol{I}_{\mathrm{GT}}, \mathcal{W}\left(\boldsymbol{I}_{\mathrm{tilt}}; \hat{\mathcal{T}}^{-1}\right)\right) \tag{5.7}$$

The final loss function is then calculated as a weighted sum of $\mathcal{L}_{char}$ and $\mathcal{L}_{tilt}$ as

$$\mathcal{L} = \alpha_1 \mathcal{L}_{char} + \alpha_2 \mathcal{L}_{tilt}. \tag{5.8}$$

where the weights α_1 and α_2 are set as 0.2 and 0.8 respectively.

The original work employed two different datasets for training: ATSyn-static and ATSyn-dynamic. For both, the data was synthesised using an improved version of the turbulence simulator from the same authors [77]. Using this, the

Table 5.4: Summary of the different configurations of DATUM trained for analysis.

Model Name	$\mathcal{L}_{char}$ α_1	$\mathcal{L}_{tilt}$ α_2	$\mathcal{L}_{percep}$ α_3
DATUM$_{CT}$	0.8	0.2	✗
DATUM$_{C}$	1	✗	✗
DATUM$_{CP}$	0.8	✗	0.012
DATUM$_{CTP}$	0.8	0.2	0.012

'tilt-only' image $\boldsymbol{I}_{\text{tilt}}$ can be generated alongside the full turbulent image.

5.4.1 Modified DATUM Models

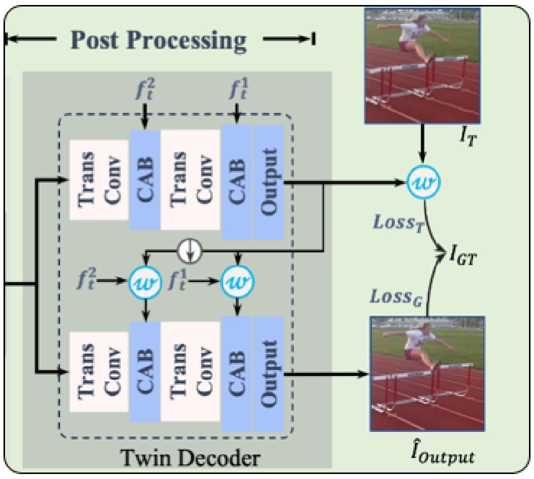
Unlike the investigation into the EDVR model, the core architecture of DATUM was left unchanged. Instead, the focus of this investigation was on the loss function. As with EDVR, this was also motivated by the use of turbulence mitigation in a larger image/video processing framework. To this end, the addition of a perceptual loss was proposed to improve the performance of DATUM.

Alongside this addition, the use of the tilt loss was also examined, as its presence requires specific data to be available (i.e. the 'tilt-only' image). Whilst this is possible via the simulator used by the original authors, such an image may not always be available, therefore restricting the training of DATUM to specific datasets. If the model was able to train without this loss, the flexibility of the model as a whole would be greatly improved.

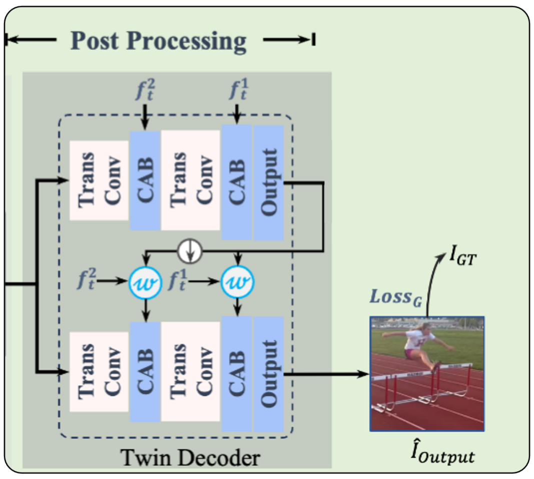
There were therefore 3 different loss functions available: The Charbonnier loss $\mathcal{L}_{char}$, the tilt loss $\mathcal{L}_{tilt}$ and the perceptual loss $\mathcal{L}_{percep}$. The final loss, $\mathcal{L}$, can therefore be represented as a weighted sum of each loss,

$$\mathcal{L} = \alpha_1 \mathcal{L}_{char}\left(\boldsymbol{I}_{\text{GT}}, \hat{\boldsymbol{I}}\right) + \alpha_2 \mathcal{L}_{tilt}\left(\boldsymbol{I}_{\text{GT}}, \boldsymbol{I}_{\text{tilt}}, \hat{\mathcal{T}}^{-1}\right) + \alpha_3 \mathcal{L}_{percep}\left(\boldsymbol{I}_{\text{GT}}, \hat{\boldsymbol{I}}\right). \quad (5.9)$$

Using different weightings of α_1, α_2 and α_3, four different configurations of the model were trained: the original DATUM configuration (DATUM$_{CT}$), a 'no tilt' configuration (DATUM$_{C}$), the original configuration with an additional perceptual loss (DATUM$_{CTP}$), and finally the 'no tilt' configuration with an additional perceptual loss (DATUM$_{CP}$). The corresponding weights for each configuration are defined in Table 5.4. For α_3, many different values were tested, examining the impact that the perceptual loss had on the total loss $\mathcal{L}$. It was found that a value of 0.012 resulted in the best results, and resulted in a perceptual loss contribution that was just above that of the Charbonnier loss $\mathcal{L}_{char}$.



(a) Original configuration of the twin decoder. With the output of the upper path being fed into the lower half, whilst also contributing towards the tilt loss alongside a 'tilt-only' version of the input.



(b) Proposed removal of the tilt loss. The output of the upper path of the twin decoder is still fed into the lower half, but not passsed to any loss function.

Figure 5.6: Comparison between the original DATUM loss configuration and the proposed removal of the tilt loss.

The Charbonnier loss and tilt loss were calculated as previously defined in Eq.s (5.2) and (5.7). The perceptual loss was implemented in the same manner as for EDVR in Section 5.3.1, where the activations were extracted from the 'conv5_4' layer of the VGG19 [120] pretrained model, and used to evaluate Eq. (5.1).

The original configuration of the Charbonnier loss and tilt loss can be seen in Figure 5.6a, where it can be seen that the inverse tilt field, $\hat{\mathcal{T}}^{-1}$, is not only passed to the tilt loss, but also fed into in the lower path of the twin decoder. This allows the lower path to focus more on blur artifacts present in the image, as the tilt is already removed. In the case of DATUM$_C$ and DATUM$_{CP}$, this connection is still implemented, however the tilt loss is no longer present in the final loss function, as shown in Figure 5.6b.

5.4.2 Training Procedure

In order to train the different configurations of the DATUM architecture, some of the training hyperparameters were altered, due to limited computation hardware. The size of the model was left unchanged, however the training length, batch size and number of frames used were reduced in order to remain within memory constraints. Each model was trained from scratch for a total of 200,000 iterations, incorporating 2,500 warm-up iterations (Reduced from 400,000 total and 10,000 warm-up). As with EDVR, the choice to train the models from scratch was to allow additional flexibility, and to not be restricted by pre-trained model architecture. In order to allow the original model size, the batch size was reduced from 8 to 4 and the number of frames provided during training was reduced from 30 to 20. As in the original work, all models were trained using a Cosine Annealing scheduler alongside an Adam optimizer and the learning rate was kept as a value of 2×10^{-4}. The training dataset used was the ATSyn-static dataset presented in the original work [88].

In order to test the different configurations, three different test datasets were utilised. Firstly the test set of the ATSyn-static dataset, which consists of images from the Places dataset [28] with turbulence distortion applied via simulation. As well as simulated data from the original author, the classification based metric, proposed in Section 5.3.1, is also used. The only difference in the case of DATUM is in the dataset size. Instead of the full 34,467 videos, a subset of 5,000 was selected. As with the experimentation with EDVR, the motivation for this dataset is the analysis of *post-mitigation classification accuracy*, where the classification is evaluated by the EfficientNetV2-L classification model.

As well as this newly proposed metric using 'turbulent imagenet', another application based metric in the form of text classification was also possible. This was due to the existence of the Turbulence Text Dataset [60]. This is a dataset consisting of real turbulent scenes, each comprising of 100 video frames of a piece of paper with five printed words. Using this dataset, similar to that of 'turbulent imagenet', the post-mitigation task of text classification can be performed. This dataset is especially important, as the distortion present is wholly due to a real-life turbulent atmosphere, and is therefore a more realistic scenario. As in the original work [60], the evaluation of post-mitigation text classifications was achieved with three different classifiers: CRNN [243], DAN [244] and ASTER [245].

In order to evaluate image quality, the PSNR, SSIM and MANIQA metrics were used for this investigation alongside the classification results.

5.4.3 Comparison with EDVR

As well as the cross-comparison of the different DATUM configurations, a comparison can also be made with the previously implemented EDVR architecture. In order to facilitate this and ensure a fair comparison, each model was trained using the same dataset. Specifically, they were trained with 20 RGB input frames from the ATSyn-static dataset [88], and tested with the corresponding test set from ATSyn-static alongside the Turbulence Text Dataset [60].

Table 5.5: Detail of the different future frames (ff) and past frames (pf) used, and the corresponding input/output frame count.

Model	ff/pf	Frames In	Frames Out
DATUM	0	15	15
DATUM	2	15	11
DATUM	7	15	1

As for the architectures, EDVR was trained with its original number of parameters and residual blocks, however, as with the models trained in Section 5.3.1, the deformable convolution layers were replaced with the proposed solution. This therefore matched the configuration of $\text{E}\overline{\text{D}}\text{VR}$ from Section 5.3.1. As for the DATUM architecture, in order to comply with hardware limitations (in line with Section 5.4.1), the batch size was reduced from 8 to 4 (DATUM_{CT}). However, unlike previous configurations, this model was given a full 400,000 training iterations. Both models were trained with their default loss functions.

Alongside the analysis of turbulence mitigation quality, the computation requirements of the two models was also compared. For this analysis, the parameter count, inference time, and GPU memory usage were evaluated.

In addition to the base comparison between $\text{E}\overline{\text{D}}\text{VR}$ and DATUM_{CT}, both methods were also evaluated for scalability with respect to input frame count and image size. The evaluation of frame count was tested by increasing the number of input frames from 10 to 60, in increments of 10, where, for each, the profiling was performed with images sizes of 256×256 and 512×512. For the image size analysis, resolution was increased from 256×256 to 1792×1792 in increments of 256, whilst the number of input frames was fixed at 15.

Finally, the use of the future/past frames (ff/pf) parameter in DATUM was also examined. This value alters the number of frames fed into the MTCSA module and dictates the number of output frames from the model. The larger this value, the more neighbouring frames are utilised for each output frame, therefore resulting in fewer total output frames. The tested values of ff/pf are shown in Table 5.5, where the resulting number of output frames is also provided.

For each test described, the profiling was evaluated 50 times, and the average was then found. It is worth noting that for all the profiling performed, only a single video was processed at a time, and was performed with the model in evaluation mode and gradient computation turned off. Therefore, whilst this analysis may indicate the potential hardware performance of the models at test time, such performance may not be applicable during training due to the added

Table 5.6: Average PSNR and SSIM results of each network. Each test dataset contained 1000 images. (PSNR/SSIM). (Best results in **bold**).

Network	Low	Medium	High
Turbulent Input	23.36/0.669	*20.90/0.544	*20.19/0.504
BRDNet	19.08/0.598	18.39/0.491	17.87/0.451
RDN	**23.77**/**0.704**	20.66/**0.568**	20.13/**0.534**
SuperSR	22.74/0.685	20.51/0.547	19.93/0.508
CAE-Unet-CAE	22.55/0.676	20.55/0.551	19.97/0.513
RCAN	22.14/0.688	19.75/0.545	19.21/0.503
DnCNN	23.35/0.669	**20.89**/0.545	**20.19**/0.505

* identifies cases where the turbulent input images actually achieved the best metric score

memory requirements.

5.5 Results

5.5.1 Single frame Turbulence Mitigation

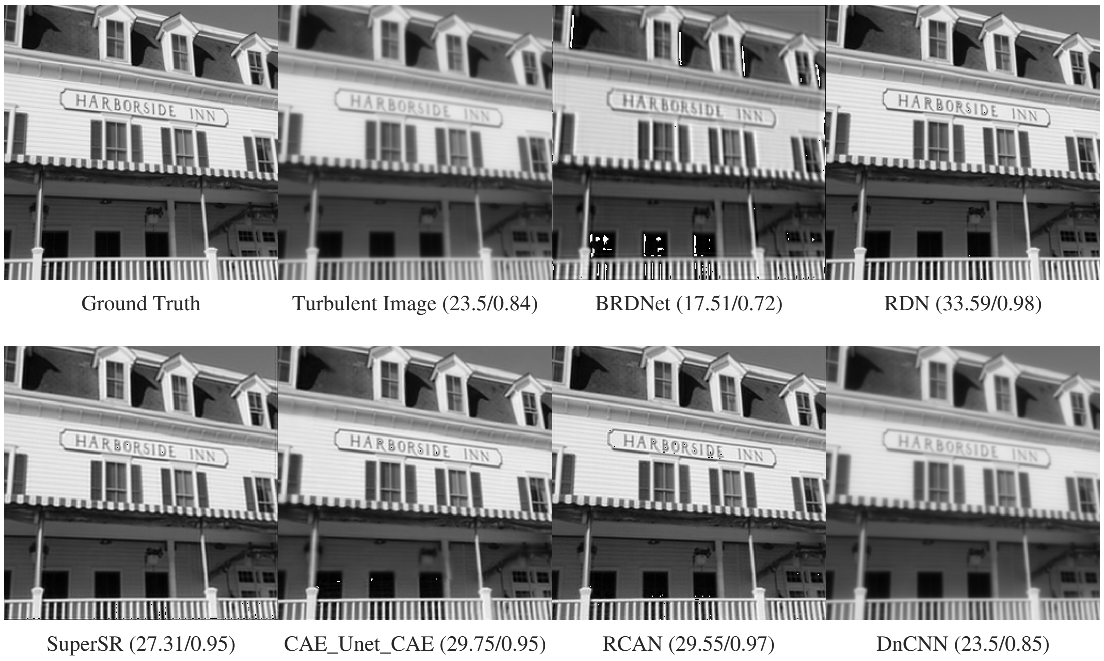
To test the single frame networks, three datasets were used: 'Low', 'Medium' and 'High'. Each comprising of 1,000 images, and varied strengths of turbulence. Once tested, the average PSNR and SSIM scores were evaluated for each model, and are shown in Table 5.6, alongside the metrics scores for the original turbulent input images.

These results show that RDN provided the highest SSIM score over all three datasets. It also achieved the highest PSNR score for the 'Low' dataset with DnCNN achieving the best PSNR scores for the 'Medium' and 'High' datasets. This therefore indicates that, of the chosen networks, it was RDN that would be best suited for turbulence mitigation. However, if the metric scores for the turbulent *input* images are also examined, it can be seen that although RDN provides the best performance out of the different models, its scores are still very close to, if not less than, those obtained from the original turbulent input, suggesting that little to no improvement was made. In order to further analyse these results, output images can be examined. Firstly, shown in Figure 5.7, are two cases where a selection of the models were successfully able to improve upon the turbulent input image, for both PSNR and SSIM. Before discussing the quantitative improvements, it is worth examining the characteristics of the turbulent input images themselves. The input shown in Figure 5.7a does not fully resemble a typical turbulent image, as it lacks clear tilt effects and appears primarily blurred. Similarly, the input in Figure 5.7b does not accurately depict a turbulent atmosphere, with dark pixel artifacts appearing disconnected from the main building. These shortcomings are a result of the simulator's development timeline where, at the time of this investigation, only the first version of the simulator described in Chapter 4 was available. Whilst these inaccuracies were addressed in the subsequent iteration of the simulation tool, the images presented here reflect the dataset that was used during this study.

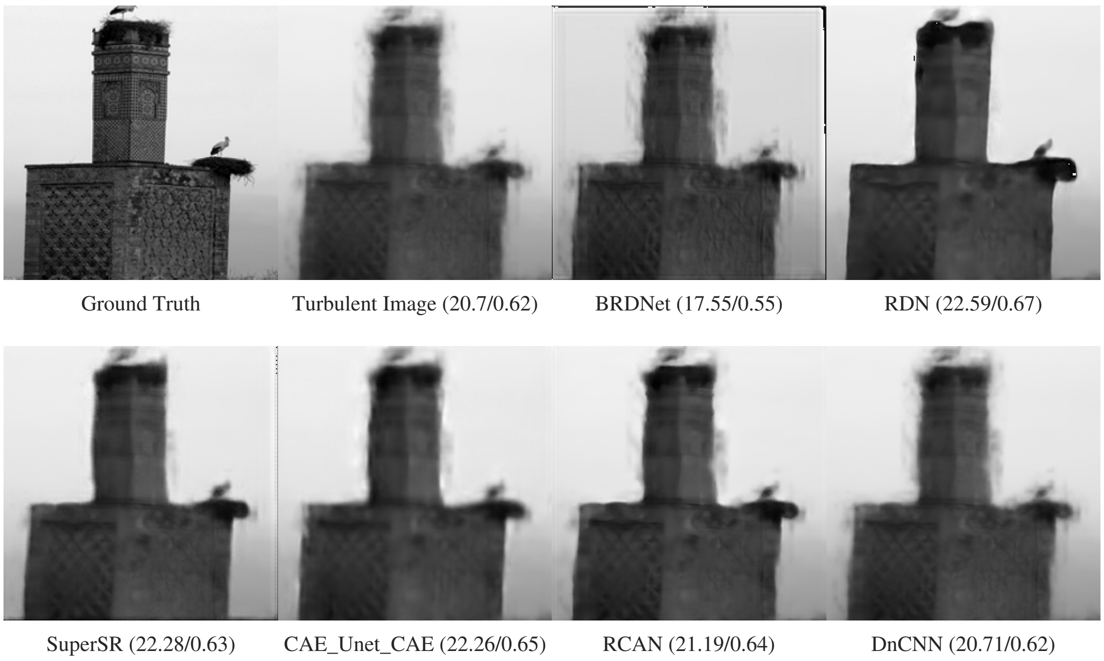
Whilst the average scores reported in Table 5.6 suggest that the models are not able to fully mitigate the effects of turbulence, these example images show that improvement is indeed possible in some cases. With the best performing model being RDN where, in the case of 'Low' turbulence, it is able to recover a high level of sharpness. For the case of 'Medium' turbulence, whilst the high frequency details are unable to be recovered, the RDN network is still able to improve the quality of the edges by a significant margin. Given these positive examples, it is also worth examining cases where the model metric scores were worse than that of the turbulent input. Such examples are shown in Figure 5.8.

Analysing these images clearly reveals why the average metric scores reported in Table 5.6 were so low. It can be seen that the image backgrounds show signs of severe clipping, and is therefore the likely reason for the poor average PSNR and SSIM scores. Despite this however, it can be seen that the areas of interest within these images have indeed been improved, where lines have been straightened and the contrast has been improved. This therefore is the first indication that commonly used metrics are unable to present an accurate description of image quality, as they are not nuanced enough to capture local improvements that may have been made.

Given this however, an analysis of these images can also be completed via a post-processing step that aims to remove the effects of clipping seen in Figure 5.8. Whilst this problem would ideally be overcome during turbulence mitigation, via a clamping or softmax operation, the only method of fixing such clipping after the fact is by comparing with the original ground truth image. By comparing the pixel values between the model outputs and the original image, any pixel that has been clipped or wrapped can be identified and fixed. This operation can be

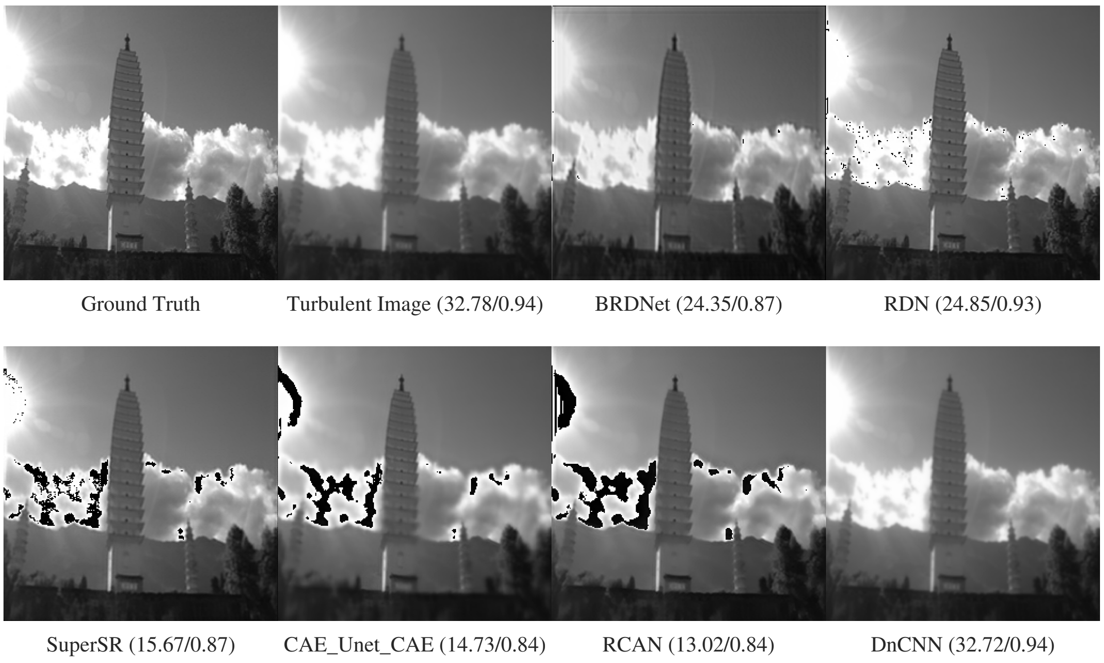


(a) Examples of successful outputs from models presented with the 'Low' turbulence sub-dataset (PSNR/SSIM)

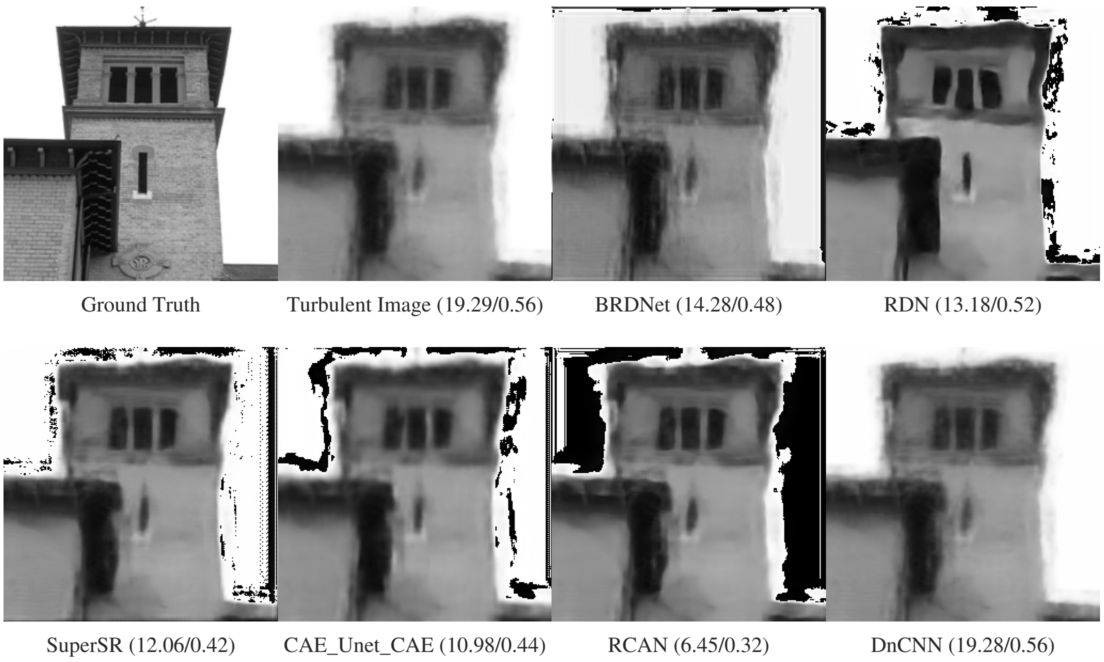


(b) Examples of successful outputs from models presented with the 'Medium' turbulence sub-dataset (PSNR/SSIM)

Figure 5.7: Example outputs of models given different levels of turbulent images. Shown are cases where at least one or more model is able to achieve higher metric scores than the turbulent input.



(a) Examples of unsuccessful outputs from models presented with the 'Low' turbulence sub-dataset(PSNR/SSIM)



(b) Examples of unsuccessful outputs from models presented with the 'High' turbulence sub-dataset (PSNR/SSIM)

Figure 5.8: Example outputs of models given different levels of turbulent images. Shown are cases where the models are unable to achieve higher metric scores than the turbulent input.

Table 5.7: PSNR and SSIM results for post-processed images using threshold values of $T_L = 20$ and $T_U = 235$. Each dataset contains 1000 images. Format: PSNR/SSIM.

Network	**Low**	**Medium**	**High**
Turbulent Input	23.36/0.669	20.90/0.544	20.19/0.504
BRDNet	20.18/0.609	18.96/0.496	18.53/0.457
RDN	**24.96**/**0.710**	21.09/**0.572**	**20.62**/**0.539**
SuperSR	23.97/0.692	20.91/0.551	20.31/0.512
CAE-Unet-CAE	24.11/0.685	**21.17**/0.557	20.51/0.519
RCAN	24.53/0.703	20.95/0.558	20.32/0.516
DnCNN	23.35/0.669	20.89/0.545	20.19/0.505

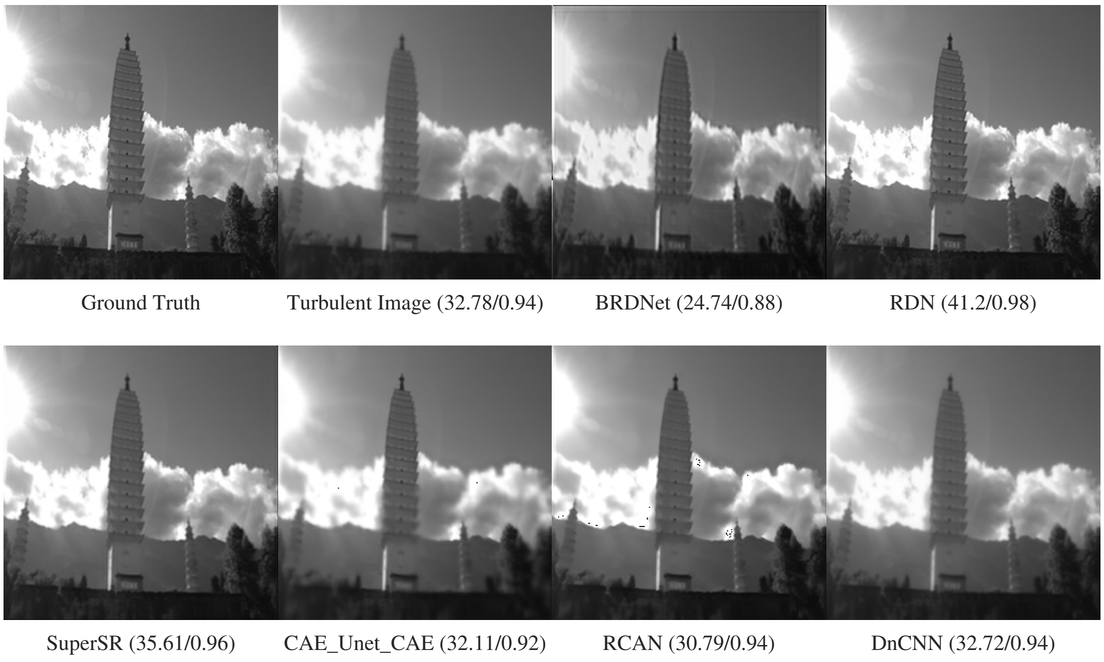
described by the following equation,

$$\boldsymbol{I}_{\text{Out}}(x,y) = \begin{cases} 255, & \text{if } \boldsymbol{I}_{\text{GT}}(x,y) > T_U \text{ and } \boldsymbol{I}_{\text{Clipped}}(x,y) < T_L \\ 0, & \text{if } \boldsymbol{I}_{\text{GT}}(x,y) < T_L \text{ and } \boldsymbol{I}_{\text{Clipped}}(x,y) > T_U \\ \boldsymbol{I}_{\text{Clipped}}(x,y), & \text{otherwise} \end{cases}, \tag{5.10}$$

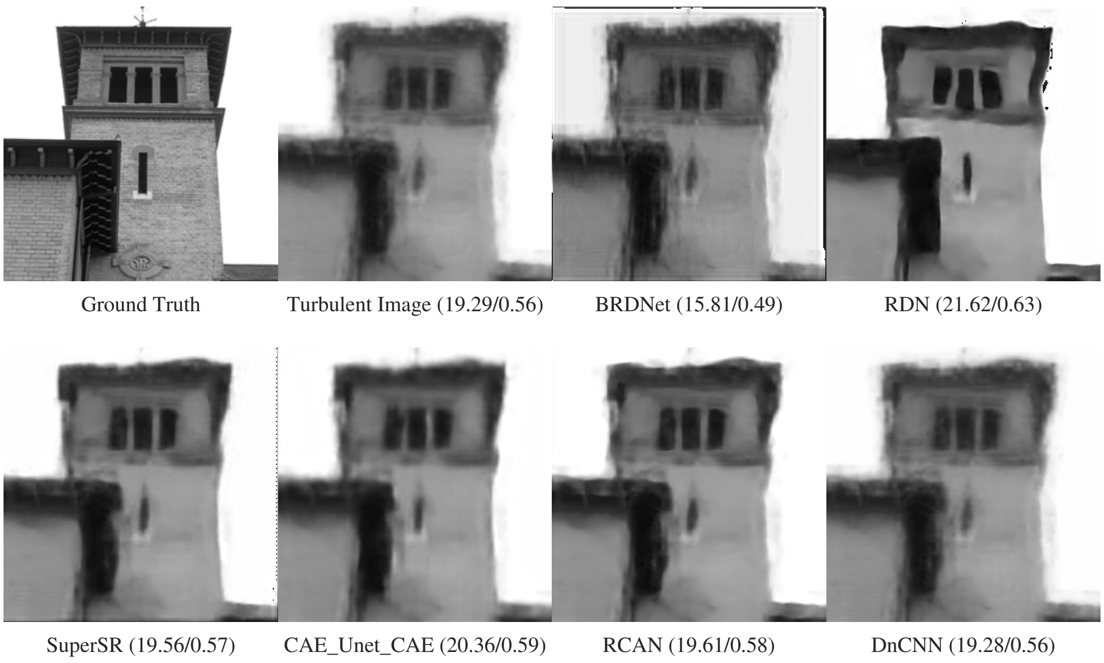
where $\boldsymbol{I}_{\text{Clipped}}(x,y)$, $\boldsymbol{I}_{\text{GT}}(x,y)$ and $\boldsymbol{I}_{\text{Out}}(x,y)$ are the clipped, ground truth and output (fixed) images respectively. The two thresholds T_L and T_U denote the sensitivity of the operation, where $T_U = 255 - T_L$ and T_L was set to a value of 20. Example outputs of this operation can be seen in Figure 5.9, where it can be seen that the clipping has been greatly reduced. The resulting metric scores on these processed images can be seen in Table 5.7.

Unlike the previous results (Table 5.6), these results show that, after post-processing, the image are indeed of better quality than the turbulent input. As before, it is RDN that provides the best results for all datasets, with the exception of PSNR for the 'Low' dataset, which is obtained by CAE-Unet-CAE. However, RDN is still very close to this highest value. Alongside quantative analysis, the performance of the models can also be judged based on visual observation. In this, it is clear that the RDN network that produces the cleaner output images, with SuperSR providing the next best results. The outputs of CAE-Unet-CAE and RCAN are visually similar to each other, and present slight improvements over the input. The outputs from BRDNet and DnCNN contain the same level of distortion as the input, therefore suggesting that they have not learnt any form of turbulence mitigation.

Whilst all models use different design techniques, RDN and SuperSR both make connections in a dense manner i.e. each layer is connected to all that fol-



(a) Examples of corrected outputs from models presented with the 'Low' turbulence sub-dataset(PSNR/SSIM)



(b) Examples of corrected outputs from models presented with the 'High' turbulence sub-dataset (PSNR/SSIM)

Figure 5.9: Results of post-processing operation to remove the effects of clipping. Compared to Figure 5.8, it can be seen that the image backgrounds have been corrected.

low. As this is the core feature of the best model (RDN) it is therefore proposed that such a design will result in improved turbulence mitigation performance. The architectures of CAE-Unet-CAE and RCAN are clearly able to make improvements over the turbulent input, but lack the representation power to fully recover sharp details. Finally, due to their simplicity, BRDNet and DnCNN are clearly not transferable for turbulence mitigation. This is due to their architecture's lacking capacity to learn the extremely complex nature of turbulence distortion.

These results provide further proof that the application of deep learning is indeed viable for the task of turbulence mitigation, provided that a suitable architecture is chosen. They indicate that models originally designed for other image processing tasks, such as deblurring or denoising, may be repurposed for other data contents without requiring changes the core architecture.

However, although this investigation provided positive results, it was deemed that even better results were still achievable. For example, it can be seen in Figure 5.9 that the output of RDN, whilst mitigated of serious warping, still lacks the high frequency detail that would be desired. It was believed that this shortcoming was less of an issue with the model architectures, and more due to the single frame input as, with such limited information, the performance of each model is naturally restricted.

This therefore shows that the task of single-frame turbulence mitigation is extremely difficult, reinforcing the findings of traditional turbulence mitigation techniques that utilise video sequences. Due to turbulence distortion being temporally varying, the use of video sequences greatly reduce the difficulty of turbulence mitigation, as it would allow a deep learning model to fully understand the underlying high quality latent image.

5.5.2 Video Turbulence Mitigation with EDVR

Deformable Convolution

The two models EDVR$_s$ and E$\overline{\text{D}}$VR$_s$ were evaluated using the test set of the 'turbulence videos' dataset. The resulting metric scores are reported in Table 5.8.

These show that the replacement of the deformable convolution makes very little impact to the model performance in the case of static turbulence mitigation. This is further highlighted in the table via the absolute difference column, which emphasises the model's similar performance. The term 'static' in this case refers to video sequences that contain static contents, i.e. no movement between frames. The opposite of this would be 'dynamic', where a video sequence would contain moving objects or were taken by a moving camera. Given the fact that the inclusion of the deformable convolution can lead to difficulties in the train-

Table 5.8: Results from the replacement of the deformable convolution layer (EDVR$_s$ vs E$\overline{\text{D}}$VR$_s$). Each was tested on the 'turbulence videos' test set, containing static scenes

	Metric	EDVR$_s$	E$\overline{\text{D}}$VR$_s$	Difference
FR	PSNR ↑	**30.8073**	30.8018	0.0055
	SSIM ↑	0.9197	**0.9199**	0.0002
	CW-SSIM ↑	**0.9656**	0.9655	0.0001
	RMSE ↓	7.9576	**7.9565**	0.0011
NR	BRISQUE ↓	36.049	**36.013**	0.0355
	NIQE ↓	**6.205**	6.243	0.038
	CLIPIQA ↑	0.472	**0.474**	0.002
	MANIQA ↑	0.2864	**0.2883**	0.0019

ing process, these results show that the proposed replacement structure not only mitigates such complications but also maintains model performance. Furthermore, the size of the E$\overline{\text{D}}$VR$_s$ model contains 8.5% fewer trainable parameters than EDVR$_s$ (3,771,393 vs 4,122,465), therefore providing further motivation to replace the DCN layers.

It is worth noting however that the data used in this test was static, and therefore may not require as much alignment as the data used in the original work. This may be a reason for the results presented, where as, if dynamic data was used, the model may require more complex alignment, which deformable convolution could assist with. Another factor that may impact the usefulness of the DCN layer is that of turbulence strength, where it may be the case that the technique might prove more useful for more difficult scenarios. Shown in Figure 5.10 are the average metric scores over a range of Low ($C_n^2 = 0.25 \times 10^{-15}$) to High ($C_n^2 = 0.25 \times 10^{-15}$) turbulence strengths. Of these plots, it is only the NR metrics that indicate any clear difference in performance for different C_n^2 values. Of these, it is only NIQE that reports EDVR$_s$ outperforming E$\overline{\text{D}}$VR$_s$ at all strength levels (as expected from Table 5.8). Whilst MANIQA reports that the two models drop in quality at the same rate, BRISQUE and CLIP_IQA indicate that the higher levels of turbulence do indeed benefit from the additional DCN layer, with BRISQUE reporting that, for the highest level of turbulence, the EDVR$_s$ model provides the best results. This indicates that the use of deformable convolutions may indeed lead to better performance when dealing with higher distortions levels, especially if paired with the additional complexity of dynamic data. However, when dealing with lower levels turbulence and static scenes, the DCN layer may not be necessary.

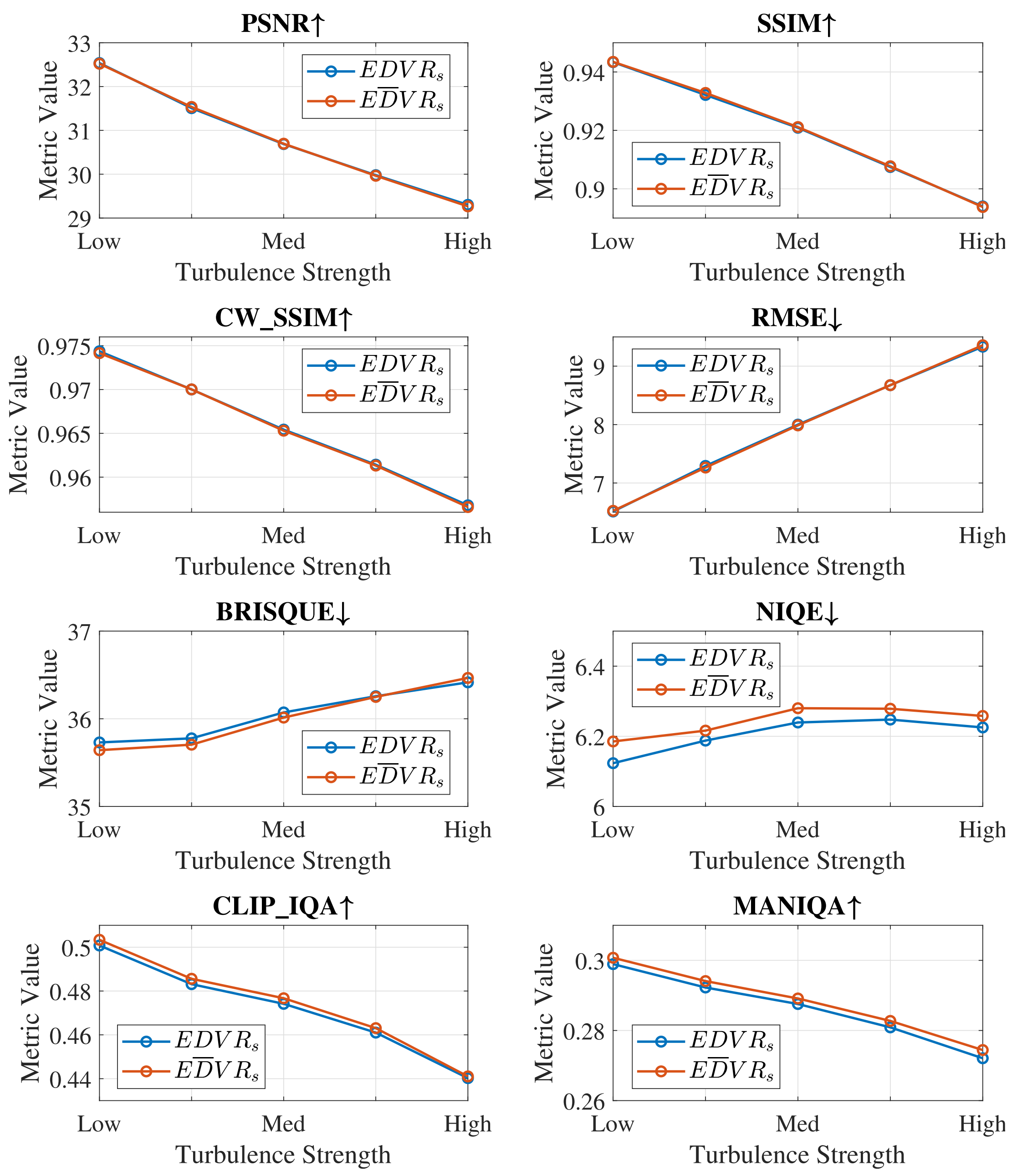


Figure 5.10: Metric scores for $\mathrm{E\overline{D}VR}_s$ and EDVR_s outputs over different turbulence strengths. Low $C_n^2 = 0.25 \times 10^{-15}$ and High $C_n^2 = 0.25 \times 10^{-15}$

As well as investigating the *replacement* of the DCN layer, the practice of clipping its learnable offsets was also tested. Table 5.9 presents the results for the EDVR_s model with four different values of clipping threshold, as well as the results of EDVR_s without clipping.

These results show that none of the clipped models are able to outperform the unconstrained version, however the performance decrease is extremely minimal, with a threshold of 5 providing the best results. This therefore indicates that the clipping does indeed impact the training performance in a small manner, and

Table 5.9: Results of EDVR$_s$ with different clipping thresholds, tested on the 'turbulence videos' test set.

	Metric	**Clipping Threshold**				
		1	**2**	**5**	**10**	**None**
FR	PSNR ↑	29.848	29.929	30.091	29.966	**30.808**
	SSIM ↑	0.905	0.906	0.909	0.908	**0.920**
	CW-SSIM ↑	0.960	0.960	0.962	0.961	**0.966**
	RMSE ↓	8.830	8.787	8.615	8.739	**7.958**
NR	BRISQUE ↓	36.070	36.300	36.300	36.388	**36.049**
	NIQE ↓	6.260	6.226	6.215	6.284	**6.205**
	CLIPIQA ↑	0.451	0.465	0.462	0.458	**0.472**
	MANIQA ↑	0.281	0.282	0.283	0.284	**0.286**

Table 5.10: Statistical values of the offset vector at the end of training from EDVR$_s$ at different clipping thresholds.

Clipping	**Offset Statistics**			
Threshold	**Min**	**Average**	**Max**	**std**
1	-1	0.0077	1	0.80
2	-2	0.026	2	1.50
5	-5	-0.0255	5	2.46
10	-9.38	0.11	9.12	2.83
None	-7.27	0.12	7.01	2.46

could potentially be a larger issue in different circumstances. To fully understand how much clipping is occurring, the statistics of the offset values after training are reported in Table 5.10.

These statistics show that, as expected, the average value of the offsets in all models is close to zero, therefore indicating that the distribution of the weights is centred about zero. It can also be seen that for the case of a threshold of 10, clipping was not actually implemented, as the min/max of the weights did not reach this threshold. It is the standard deviation of the final weights that is the most informative element of these results. For the model with a clipping threshold of 1, the std was 0.8. This is 80% of the threshold value, implying that the distribution of the weights were heavily restricted by the clipping procedure and would have grown larger if given the opportunity. The results from the higher thresholds prove this as, once a threshold above 2 was used, the std of the weights seemed to 'settle' at a value between 2 and 3. This suggests that a threshold of 5 would be the preferred value if clipping were to be implemented, as this allows enough space for the offsets to work with, whilst also not allowing them to grow

Table 5.11: Classification accuracy of the 'turbulent imagenet' dataset after turbulence mitigation by $\mathrm{E\overline{D}VR}_{s25}$, $\mathrm{E\overline{D}VR}_{s50}$ and EDVR_s. Classification accuracy on the turbulent input was **74.48%**.

Model	**Charbonnier Weight** (λ)	**Perceptual Weight** (α)	**Classification accuracy**	**Increase**
$\mathrm{E\overline{D}VR}_s$	1	0	90.14%	+15.66%
$\mathrm{E\overline{D}VR}_{s25}$	0.75	0.25	**92.34%**	**+17.86%**
$\mathrm{E\overline{D}VR}_{s50}$	0.5	0.5	92.33%	+17.85%

too large. This is also the threshold that provided the best results.

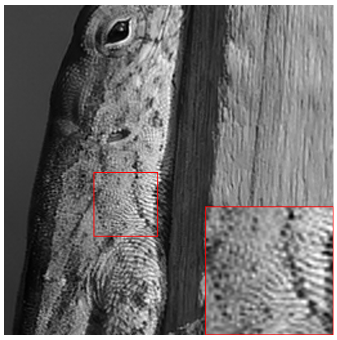
However, if the 'settled' std of the weights were to increase (e.g. via architecture or data changes), a threshold of 5 may result in more clipping, and therefore reduced performance. In such a case, choosing a larger threshold of 10 may be more appropriate; however, a key caveat is that if the offsets reach such high values, it likely indicates that training has already become unstable, and clipping alone will not be sufficient to prevent it. The choice of the threshold is therefore a very delicate decision. Too small, and performance will be limited. Too large, and the clipping of the weights may come into effect too late, as the model has already become unstable.

Perceptual Loss

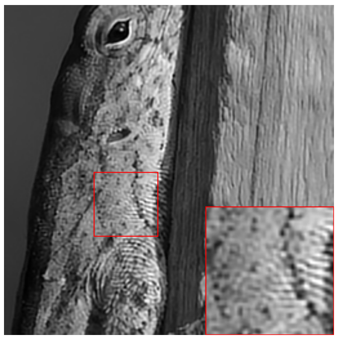
It was proposed that the introduction of a perceptual loss to the training of $\mathrm{E\overline{D}VR}_s$ would result in improved post-processing classification. Each of the proposed models ($\mathrm{E\overline{D}VR}_s$, $\mathrm{E\overline{D}VR}_{s25}$, $\mathrm{E\overline{D}VR}_{s50}$) were evaluated on the 'turbulent imagenet' test dataset and the resulting turbulence mitigated images were passed to the EfficientNetV2-L classification model. Each video in the 'turbulent imagenet' dataset contains 15 frames and, as detailed in Chapter 4, were preprocessed prior to this evaluation, where each of ground truth images within the final dataset are correctly classified by the EfficientNetV2-L model. This pre-processing removes any losses that may have been incurred during cropping and grayscale conversion. The resulting accuracy for each EDVR model is reported in Table 5.11.

The first highlight of these results is the fact that for all models, regardless of the loss function, classification accuracy is improved following turbulence mitigation, therefore showing that the use of deep learning for turbulence mitigation is an extremely useful tool for such post-processing tasks.

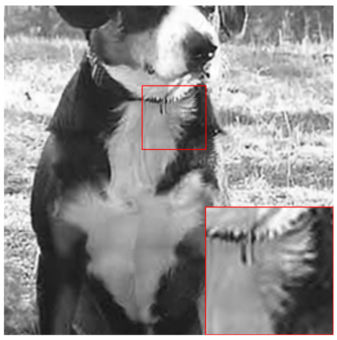
With regard to the presence of the perceptual loss, it is clear that models trained with such an addition are indeed able to outperform those trained with



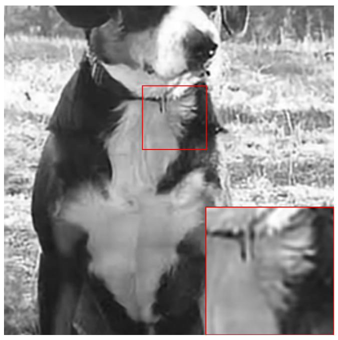
(a) Output image from $\mathrm{E\overline{D}VR}_{s25}$. Correctly predicted as: ‘American chameleon’



(b) Output image from $\mathrm{E\overline{D}VR}_{s}$. Incorrectly predicted as: ‘Whiptail lizard’



(c) Output image from $\mathrm{E\overline{D}VR}_{s25}$. Incorrectly predicted as: ‘Greater Swiss Mountain dog’



(d) Output image from $\mathrm{E\overline{D}VR}_{s}$. Correctly predicted as: ‘EntleBucher’

Figure 5.11: Two cases of classification performance. Top: A case where $\mathrm{E\overline{D}VR}_{s}$ outperforms $\mathrm{E\overline{D}VR}_{s25}$. Bottom: A case where $\mathrm{E\overline{D}VR}_{s25}$ outperforms $\mathrm{E\overline{D}VR}_{s}$

the original loss function.

This was the desired outcome, but was also as expected. The perceptual loss is, by definition, the use of a classifier’s weights to determine image quality. Given the minimisation of such a loss, it is reasonable to presume that the resulting images from such a model would therefore be easier to classify. This does not however guarantee that the images are of better quality. With such a dependence on classification, the model may have learnt to simply generate classifiable images, that have no bearing on the original input. i.e. the model has ‘hallucinated’ its output image. This is the reason that the perceptual loss is not used in isolation, but rather alongside more traditional loss functions. The combination of losses allows the models to maintain the overall image structure, whilst also promoting sharp classifiable features. To check if this is indeed the case, the output images can be visualised and assessed for any such ‘hallucinations’.

Shown in Figure 5.11 are two different scenarios: firstly, a case where $\mathrm{E\overline{D}VR}_{s}$

Table 5.12: Count of cases where MANIQA (NR) agrees/disagrees with PSNR (FR).

PSNR	MANIQA	Image count	Percentage
$\mathrm{E\overline{D}VR}_{s50} < \mathbf{E\overline{D}VR}_{s}$	$\mathrm{E\overline{D}VR}_{s50} < \mathbf{E\overline{D}VR}_{s}$	1131	3.28%
$\mathrm{E\overline{D}VR}_{s50} < \mathbf{E\overline{D}VR}_{s}$	$\mathbf{E\overline{D}VR}_{s50} > \mathrm{E\overline{D}VR}_{s}$	31523	91.46%
$\mathbf{E\overline{D}VR}_{s50} > \mathrm{E\overline{D}VR}_{s}$	$\mathrm{E\overline{D}VR}_{s50} < \mathbf{E\overline{D}VR}_{s}$	82	0.24%
$\mathbf{E\overline{D}VR}_{s50} > \mathrm{E\overline{D}VR}_{s}$	$\mathbf{E\overline{D}VR}_{s50} > \mathrm{E\overline{D}VR}_{s}$	1731	5.02%

obtains a correct classification and $\mathrm{E\overline{D}VR}_{s25}$ does not, and secondly a case where $\mathrm{E\overline{D}VR}_{s25}$ results in correct classification and $\mathrm{E\overline{D}VR}_{s}$ does not.

Firstly, from visual inspection of these images, it can be seen that the images are clear of any 'hallucinations' induced by the perceptual loss therefore suggesting that the improved classification accuracy is not due to unwanted artifacts, but because of the models ability to promote sharper images with higher detail. This can be seen in both scenarios, where the perceptual quality of the images generated by the $\mathrm{E\overline{D}VR}_{s25}$ are sharper than those from the $\mathrm{E\overline{D}VR}_{s}$ model.

Given this added sharpness however, it can be seen in Figures 5.11c and 5.11d, that the output of the $\mathrm{E\overline{D}VR}_{s}$ model is successfully classified, whilst the $\mathrm{E\overline{D}VR}_{s25}$ output is not. This is a rare case of where the added sharpness does not aid in classification accuracy. However, as shown by the average results in Table 5.11, cases like this are in the minority. Furthermore, if the classification was to be performed by a human, the added sharpness provided by $\mathrm{E\overline{D}VR}_{s25}$ would not make a significant difference to classification accuracy. However, if such a task were to be autonomous, the increased confidence in classification accuracy provided by a perceptual loss model would be invaluable.

It is worth highlighting the similarity between the two perceptual loss models. The reason for this is the difference in scale between the two losses. During training, typical values of the Charbonnier loss are $\mathcal{L}_{char} \approx 0.04$, whereas typical values of perceptual loss are $\mathcal{L}_{percep} \approx 1$. Therefore, for both ratios of λ/α, the Charbonnier loss $\mathcal{L}_{char}$ only represents a small magnitude of the total loss, $\mathcal{L}$. This results in both models being predominantly controlled by the perceptual loss element, hence their similar performance. It is positive therefore that such a dominance of the perceptual loss did not lead to hallucinations, and that the Charbonnier loss was still able to contribute towards the output image structure.

Finally, these results also show the importance of using application based tasks, such as classification, as a measure of model quality, alongside more common image quality metrics. Table 5.12 shows that, for 91.46% of the dataset, the PSNR metric would have identified $\mathrm{E\overline{D}VR}_{s}$ as the better model, while the

Table 5.13: Results of $\text{E}\overline{\text{D}}\text{VR}_s$ and $\text{E}\overline{\text{D}}\text{VR}$, tested on the ‘turbulence videos’ test set.

	Metric	$\textbf{E}\overline{\textbf{D}}\textbf{VR}_s$	$\textbf{E}\overline{\textbf{D}}\textbf{VR}$
FR	PSNR↑	30.802	**32.763**
	SSIM↑	0.920	**0.943**
	CW-SSIM↑	0.966	**0.974**
	RMSE↓	7.956	**6.356**
NR	BRISQUE↓	**36.013**	36.311
	NIQE↓	6.243	**6.236**
	CLIPIQA↑	0.474	**0.497**
	MANIQA↑	0.288	**0.300**

Table 5.14: Classification accuracy of the ‘turbulent imagenet’ dataset after turbulence mitigation by EDVR_s and $\text{E}\overline{\text{D}}\text{VR}$. Classification accuracy on low resolution input was **74.48%**.

Model	**Charbonnier Weight** (λ)	**Perceptual Weight** (α)	**Classification accuracy**	**Increase**
$\text{E}\overline{\text{D}}\text{VR}_s$	1	0	90.14%	+15.66%
$\text{E}\overline{\text{D}}\text{VR}$	1	0	**91.62%**	**+17.14%**

$\text{E}\overline{\text{D}}\text{VR}_{s25}$ obtained a classification accuracy of 92.34%

MANIQA metric would have identified $\text{E}\overline{\text{D}}\text{VR}_{s50}$. Given that that the outputs of $\text{E}\overline{\text{D}}\text{VR}_{s50}$ lead to increased classification accuracy, this indicates that the MANIQA metric would be far more reliable in identifying ‘good images’ for classification. In such a case, the PSNR metric would be extremely misleading, as it clearly prefers the less sharp images produced by $\text{E}\overline{\text{D}}\text{VR}_s$. These results highlight a significant issue with the commonly used metrics in the literature, in that they are not always able to identify the best performing models; this issue is further explored in Chapter 6.

Model Size and Ablation Study

As well as the alteration of the fundamental architecture and training procedure, the $\text{E}\overline{\text{D}}\text{VR}$ model was also tested in regards to its size alongside an ablation study of its modules. Firstly, the result of increasing the model size is shown in Table 5.13.

With a total of 23,519,873 learnable parameters (compared to 3,771,393 within $\text{E}\overline{\text{D}}\text{VR}_s$) it was as expected that the larger model, $\text{E}\overline{\text{D}}\text{VR}$, was able to provide better results. Whilst this is the case, if the deployment of such a model was

Table 5.15: Performance Comparison of EDVR with and without the TSA module (PSNR/MANIQA).

Model	w/o TSA	With TSA	Percentage Increase
PCD_PD	30.470/0.287	**30.706**/**0.290**	0.773/0.959%
PCD	30.766/**0.293**	**30.859**/0.290	0.301/-1.497%
PD	29.203/0.276	**29.430**/**0.279**	0.777/0.923%
None	29.105/**0.280**	**29.139**/0.275	0.117/-2.017%

Table 5.16: Performance Comparison of EDVR with and without the PCD module (PSNR/MANIQA).

Model	w/o PCD	With PCD	Percentage Increase
TSA_PD	29.430/0.279	**30.706**/**0.290**	4.336/4.013%
TSA	29.139/0.275	**30.859**/**0.290**	5.903/5.333%
PD	29.203/0.276	**30.470**/**0.287**	4.339/3.977%
None	29.105/0.280	**30.766**/**0.293**	5.708/4.777%

Table 5.17: Performance Comparison of EDVR with and without the PD module (PSNR/MANIQA).

Model	w/o PD	With PD	Percentage Increase
TSA_PCD	**30.859**/**0.289**	30.706/**0.289**	-0.496/0.095%
TSA	29.139/0.275	**29.430**/**0.279**	1.000/1.365%
PCD	**30.766**/**0.293**	30.470/0.287	-0.962/-2.340%
None	29.105/**0.280**	**29.203**/0.276	0.337/-1.588%

hardware limited, these results show that the small model is still able to provide competitive results, even with 16% of the parameters. The $\overline{\text{ED}}$VR model can also be evaluated on the classification dataset 'turbulent imagenet' to assess the potential classification improvements from using a larger model size. The corresponding results are presented in Table 5.14.

Again, as expected, the larger model is able to provide improved results. However, even with the added parameters, $\overline{\text{ED}}$VR was still unable to outperform results of $\overline{\text{ED}}\text{VR}_{s25}$, therefore showing the relative power of model size vs loss function. If the loss function is chosen appropriately, it can compensate for a smaller model size.

Finally, the results of the ablation study can be seen in Tables 5.15, 5.16 and 5.17. For each table, the PSNR and MANIQA metrics are presented, and the modules that make the biggest contributions can be identified.

Firstly, the inclusion of the PCD module is the most noticeable result, as its

Table 5.18: FR and NR Metric results for each DATUM configuration, tested on the ATSyn-static [88] test set.

	Metric	DATUM Configurations *CT*	*CTP*	*C*	*CP*
FR	PSNR ↑	<u>25.329</u>	24.971	**25.335**	24.988
	SSIM ↑	<u>0.7477</u>	0.734	**0.7482**	0.733
NR	MANIQA ↑	<u>0.256</u>	0.243	**0.259**	0.241

C: Charbonnier Loss, **T**: Tilt Loss, **P**: Perceptual loss

presence makes a significant difference to the overall model performance. This is most likely due to the fact that the majority of the model weights exist within this module, and therefore its absence also correlates with a significant reduction in model representation ability. It is also likely that this module is responsible for the majority of the image recovery process. Whilst the other modules have their parts to play, they do so on top of the initial processing performed by the PCD module. This is also the case for the TSA module which, on the whole, also provides an improvement when present.

Another observation from these results is the impact of the PreDeblur module. It can be seen in Table 5.15 that the TSA module makes a larger contribution when the PD module is also present. However, in Table 5.16, the presence of the PD module seems to reduce the contribution from the PCD module. This is unexpected, as the purpose of the PreDeblur module is to preprocess the input to aid alignment. However, these results suggest that the PreDeblur module might be adding unnecessary processing to the images, leading to the reduced capability of the PCD module, but the increased capability of the TSA module. It seems however, that the impact to the PCD module outweighs the improvement in the TSA module, as the best performing model is that of TSA_PCD, implying that the removal of the PreDeblur would result in a better model. It is important to note however that these results only represent a single scenario, i.e. simulated grayscale turbulent images. It could be the case that the use of RGB images, or a different simulator, may result in the EDVR modules reacting differently.

5.5.3 Turbulence Mitigation with DATUM

The previous results showed that the addition of a perceptual loss to the EDVR model greatly improved its performance, and resulted in higher quality, sharper images. The following section outlines the results of a similar investigation into

Table 5.19: NR Metric results and text classification accuracies for each DATUM configuration, tested on the Turbulence Text Dataset [60]. Each configuration was tested twice: first with 20 frames as input, then with 60 frames as input.

	Metric	DATUM Configurations			
		CT	***CTP***	***C***	***CP***
Text (20) Classification	MANIQA ↑	0.314	<u>0.321</u>	0.308	**0.324**
	CRNN	86.60%	<u>86.80%</u>	**88.20%**	**88.20%**
	DAN	92.00%	93.60%	**95.00%**	<u>94.80%</u>
	ASTER	93.20%	**94.00%**	<u>93.40%</u>	92.80%
Text (60) Classification	MANIQA ↑	<u>0.342</u>	0.340	0.337	**0.379**
	CRNN	90.20%	<u>92.35%</u>	**94.10%**	85.30%
	DAN	94.00%	<u>97.50%</u>	**97.65%**	97.10%
	ASTER	94.05%	**97.55%**	96.40%	<u>97.10%</u>

C: Charbonnier Loss, **T**: Tilt Loss, **P**: Perceptual loss

the DATUM architecture, using four different configurations of the original training loss. Each configuration was tested on three separate datasets, the results of which are provided for each test dataset in turn.

Natural Scene Metrics

The first results are those from the test set of ATSyn-static, and are shown in Table 5.18, from which two important conclusions can be derived.

Firstly, the configurations that include a perceptual loss provide the lowest scores, especially for the FR metrics. This is due to the fact that these metrics do not take perceptual quality into account, and simply score the images by how similar they are to the ground truth. Therefore, any additional improvements made by the perceptual loss will be overlooked.

Secondly, and most significantly, the model's performance remains unaffected by the removal of the tilt loss and, in fact, improves. Due to the training limitation caused by the tilt loss (requirement of 'tilt-only' images) this is a significant finding, as its removal will allow any data to be used whilst also improving the performance of the model.

Text Classification

The text classification results and MANIQA scores for the Turbulence Text Dataset are shown in Table 5.19, where each configuration of the DATUM loss was tested twice. First with 20 input frames (Text (20)), and then with 60 (Text

(60)). This was to assess how the performance of each model would change when provided different levels of information.

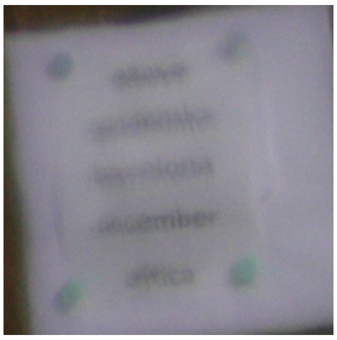
The results show that the more frames that are provided to the model, the better the mitigation performance, leading to much better classification. This is as expected for DATUM, as the increased frame count allows the feature registration and temporal fusion to better understand the distortion present.

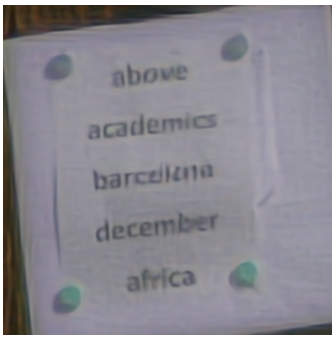
As for the comparison between DATUM loss configurations, once again it is shown that the removal of the tilt loss does not lead to reduced performance. In fact, in this case, the DATUM$_C$ model is able to outperform DATUM$_{CT}$ by a significant margin, boosting the text classification by an average of 1.6% for Text (20) and 3.3% for Text (60). A similar pattern for Text (20) can also be seen between DATUM$_{CP}$ and DATUM$_{CTP}$ where, DATUM$_{CP}$ facilitates better classification for CRNN and DAN. However, for ASTER, and Text (60), it can be seen that the removal of the tilt loss leads to a reduced classification accuracy when a perceptual loss is included.

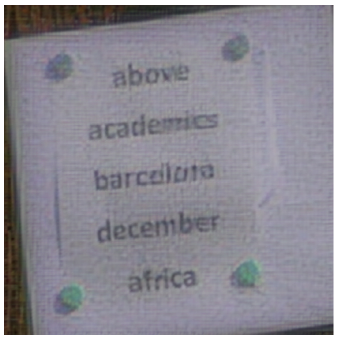
As for the impact of the perceptual loss, it can be seen that its addition to the base model provides minimal improvement to classification accuracy when provided with 20 frames, but is able to increase the effectiveness of all classifiers when provided with 60 frames, where DAN and ASTER gain significant improvement. Once the tilt loss is removed, the resulting impact of an additional perceptual loss is varied. For Text (20), the perceptual loss seems to inhibit the classification accuracy for both DAN and ASTER, whilst leaving CRNN unaffected. Whereas, for Text (60), the perceptual loss improves ASTER, whilst inhibiting CRNN and DAN.

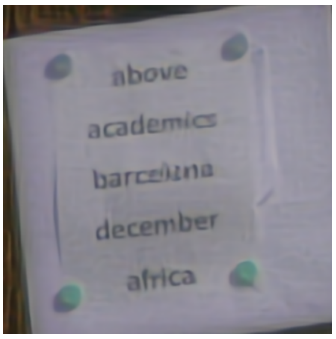
To gain a better understanding of why the different configurations provide such different classification accuracies, example output images can be analysed. Such images are shown in Figures 5.12 and Figure 5.13, with the first notable difference being the clear improvement gained by providing the model with more frames. This provides further validation of the fact that the use of video sequences can greatly increase the performance of turbulence mitigation algorithms.

The largest difference that can be seen between the outputs is in the change induced by a perceptual loss, which results in the models including a 'high frequency' texture in their images. This is due to the fact that the models were trained using the natural scene contents of ATSyn-Static so, when presented with somewhat plain images that contain minimal features, they still attempt to apply high frequency components, manifesting as noise. This is not to say that the resulting images are of bad quality however, and such an observation would be


(a) Sample input frame


(b) DATUM$_{CT}$


(c) DATUM$_{CTP}$


(d) DATUM$_{C}$

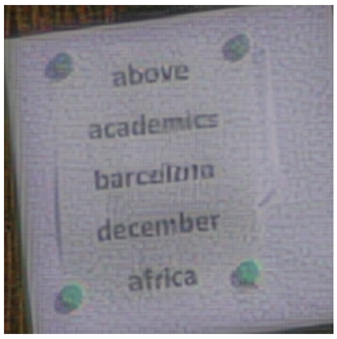
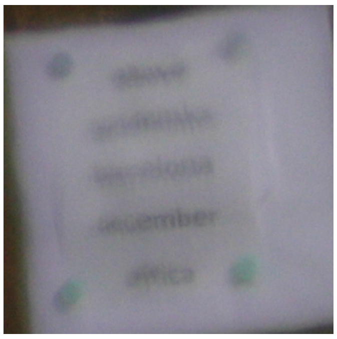
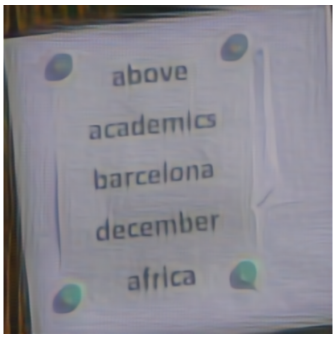
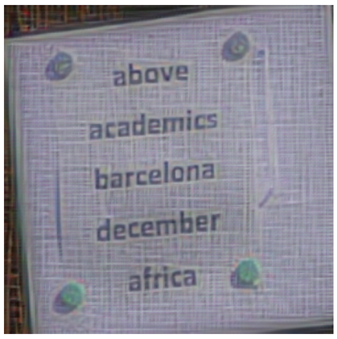
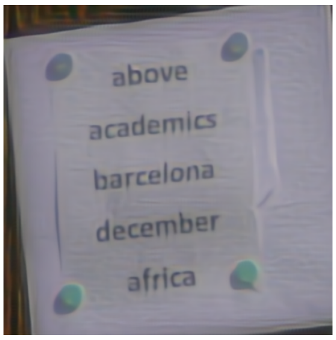

(e) DATUM$_{CP}$

Figure 5.12: Examples of turbulence mitigated images from the Turbulence Text Dataset, with 20 input frames.


(a) Sample input frame


(b) DATUM$_{CT}$


(c) DATUM$_{CTP}$


(d) DATUM$_{C}$

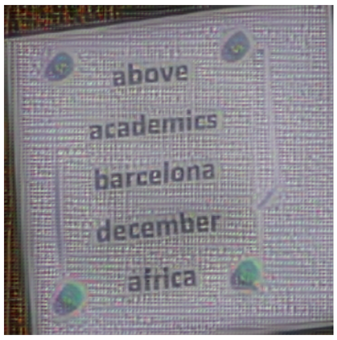

(e) DATUM$_{CP}$

Figure 5.13: Examples of turbulence mitigated images from the Turbulence Text Dataset, with 60 input frames.

dependant on the application. For example, if the turbulence was being mitigated for the images to then be used by humans, this added noise could appear as an unwanted distortion. However, the added high frequency content may even aid in some post-processing, such as text classification. In Figure 5.13c for example it could be argued that the additional noise provided by DATUM$_{CTP}$ actually increases the readability of the text, which is reinforced by the additional 3.05% average classification accuracy of this model over DATUM$_{CT}$. The opposite is true however in the comparison of DATUM$_{C}$ and DATUM$_{CP}$, where the output of DATUM$_{CP}$ has a higher level of noise (Figure 5.13e). It is clear by the results in Table 5.19 that such a high level of noise in fact reduces the capability of the text classification models. These results and images therefore show that the application of a perceptual loss must be done with caution. If testing on data that contains features outwith the training data, the models may impart unexpected distortions, such as noise, which may be beneficial in some cases, but detrimental in others.

As for the tilt loss, its presence during training results in a more subtle change to the output images. For example, comparing Figure 5.13d and Figure 5.13b, it can be seen that there is a sharper difference at edge of the paper when the tilt loss is absent. In Figures 5.12b and 5.12d it can also be seen that the absence of the tilt loss allows the model to better recreate the text contents. This can be especially noticed in the 'ber' of 'december' and the 'fri' of 'africa', and is likely the source of the average accuracy increase of 1.6%.

When the perceptual loss was present however, it was found that the accuracy of Text (60) reduced once the tilt loss was removed. By analysing the difference between Figures 5.13c and 5.13e it can be seen that the presence of the tilt loss seems to restrict the level of noise being induced by the perceptual loss. This is also the case for the Text (20) images, but is far less severe. This observation therefore implies that the interaction between the perceptual loss and the tilt loss is far more 'reactive' when more input frames are available, and that the tilt loss can be utilised to 'control' the strength of the perceptual loss.

Given the results, and the visual analysis, it is difficult to identify any one loss configuration that is best for turbulence mitigation in text scenes. However, various recommendations can still be proposed. Firstly, the more frames available to the model, the better the performance. By providing the model with as much information as possible, it is better able to recreate the high quality latent image. Secondly, if providing only 20 frames to the model, the presence of the tilt loss is not necessarily required as, in all cases, its removal provides increased

Table 5.20: MANIQA results and EfficientNetV2-L object classification accuracies for each DATUM configuration, tested on the subset of 'turbulent imagenet'.

Metric		DATUM Configurations *CT*	*CTP*	*C*	*CP*
	MANIQA ↑	**0.345**	0.321	<u>0.343</u>	0.32
Classification	Top 1	84.10%	**85.12%**	84.28%	<u>84.94%</u>
	Top 5	96.96%	97.02%	<u>97.04%</u>	**97.06%**

C: Charbonnier Loss, **T**: Tilt Loss, **P**: Perceptual loss

classification accuracy, and any high noise induced by a perceptual loss is reduced by the lack of input data. Finally, if more frames are available, the ratio between the tilt loss and perceptual loss requires careful consideration. If not using a perceptual loss, then the best performance is achieved by also removing the tilt loss. However, if using the perceptual loss, the inclusion of the tilt loss allows a degree of 'control' over the tendency of the perceptual loss to induce noise into the output image.

Object Classification

The final test dataset was that of the 'turbulent imagenet' subset. The classification accuracies of the EfficientNetV2-L classifier, as well as the MANIQA metric score, are reported in Table 5.20.

Unlike the text classification results, these show a clear winner in terms of classification accuracy. The DATUM$_{CTP}$ model is able to outperform all models in top1 accuracy, with DATUM$_{CP}$ providing the second best results. This therefore clearly shows that in the case of 'turbulent imagenet', the addition of a perceptual loss will result in models that generate more classifiable images, regardless of the presence of the tilt loss. In contrast to the text classification results, the data found within 'turbulent imagenet' is closer to the training data for these models. Therefore, as the perceptual models were training, they learnt that the application of high frequency content led to a more classifiable image. Therefore, when presented with the Turbulence Text Dataset, this learnt characteristic manifested as visible noise.

As with the text dataset, the interaction between the tilt loss and the perceptual loss provides interesting results. When a perceptual loss was used, the tilt loss aided in the performance and resulted in the best overall model. This aligns with the results found for the text data, where the combination of both the tilt loss and perceptual loss lead to a strong performance. Alternatively, if a

Table 5.21: Metric results and text classification accuracies of the ATSyn-static and Turbulence Text Dataset after turbulence mitigation by $\text{E}\overline{\text{D}}\text{VR}$ and DATUM_{CT}.

	Metric	**$\text{E}\overline{\text{D}}\text{VR}$**	**DATUM_{CT}**
ATSyn-static	PSNR ↑	25.014	**25.405**
	SSIM ↑	0.728	**0.751**
	MANIQA ↑	0.246	**0.262**
Text (20) Classification	CRNN	**88.80%**	86.00%
	DAN	**92.80%**	92.40%
	ASTER	**94.40%**	93.60%
	MANIQA ↑	0.269	**0.294**

perceptual loss is not used, adding the tilt loss results in reduced performance.

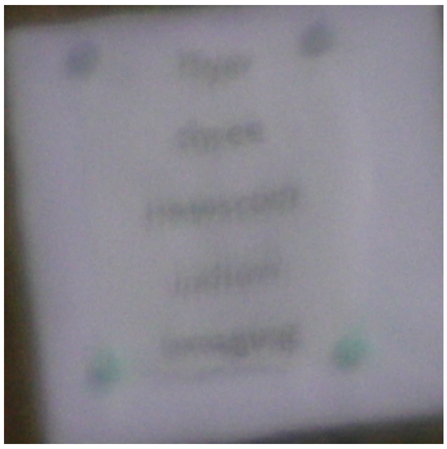
Comparison with EDVR

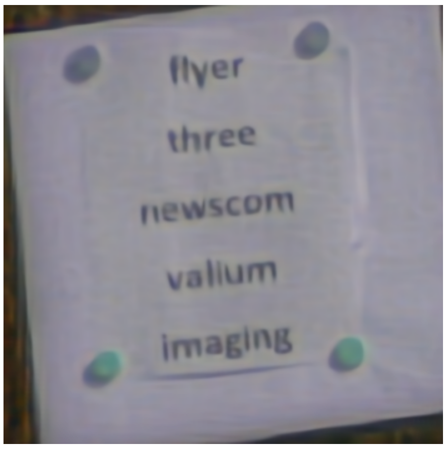
The metrics and text classification results of the model comparison between $\text{E}\overline{\text{D}}\text{VR}$ and DATUM_{CT} are reported in Table 5.21.

These results are particularly significant, as they show that $\text{E}\overline{\text{D}}\text{VR}$ can indeed compete with the more 'turbulence specific' architecture of DATUM_{CT}, despite originally being designed for other tasks. $\text{E}\overline{\text{D}}\text{VR}$ was even able to *just* outperform DATUM_{CT} in the text classification test. To further understand these results, two example output images from the Turbulent Text Dataset are shown in Figure 5.14, where it can be seen that both models are able to perform well on the given input. The output from $\text{E}\overline{\text{D}}\text{VR}$ however does seem slightly more 'natural' with the output of DATUM_{CT} having bolder text, leading to a higher contrast.

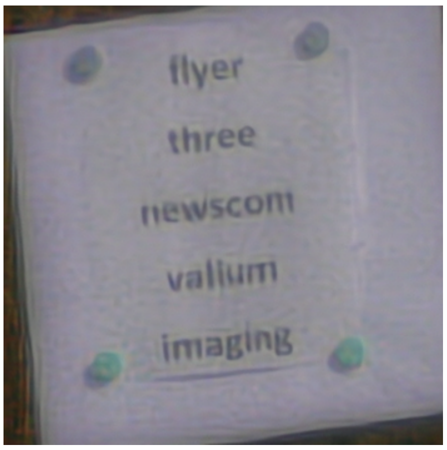
Given the similarity of the performances of the two models, other factors may also be taken into consideration, for example functionality. The values in Table 5.22 show the profiling results of the two models DATUM_{CT} and $\text{E}\overline{\text{D}}\text{VR}$. Due to the fact that DATUM_{CT} outputs more frames, its overall inference time is longer. Therefore, to ensure a fair comparison, the timing results are reported on a 'per output frame' basis. The reported memory usage refers to the peak GPU memory allocated during inference. All results were obtained using a single NVIDIA A100 SXM GPU.

It is shown that $\text{E}\overline{\text{D}}\text{VR}$ has approximately $4\times$ more parameters than DATUM_{CT}, and is slower to produces output frames. This is especially true in the case of the larger 512×512 image size, where the time-per-output of $\text{E}\overline{\text{D}}\text{VR}$ increases by 86%, compared to an increase of only 47.37% for DATUM_{CT}. However, despite these shortcomings, it can also be seen that $\text{E}\overline{\text{D}}\text{VR}$ is able to process the video se-


(a) Sample input frame



(b) DATUM$_{CT}$



(c) E$\overline{\text{D}}$VR

Figure 5.14: Example outputs from the Turbulence Text Dataset after mitigation by DATUM$_{CT}$ and E$\overline{\text{D}}$VR. Each was supplied with 20 input frames.

Table 5.22: Profiling results of the two models DATUM$_{CT}$ and E$\overline{\text{D}}$VR for two image sizes.

	Frames		**Parameters**	**256 × 256**		**512 × 512**	
	In	**Out**		**Time**	**Memory**	**Time**	**Memory**
DATUM$_{CT}$	20	16	5,779,873	19ms	1.92 GB	28ms	7.59 GB
E$\overline{\text{D}}$VR	20	1	23,687,171	50ms	1.53 GB	93ms	5.83 GB

quence whilst utilising less GPU memory, despite having more parameters. This therefore makes E$\overline{\text{D}}$VR more suitable if hardware memory is limited.

The response of each model to an increasing frame size is shown in Figure 5.15, where inference time-per-output and GPU memory allocation is shown. As expected, both time and memory usage increase with image size for both models. However it can be seen that the time for E$\overline{\text{D}}$VR is always greater than that of

DATUM$_{CT}$, and also increases at a higher rate. Nevertheless, similar to the results in Table 5.22 this behaviour is swapped in the GPU allocation, where it is DATUM$_{CT}$ that requires more memory, and scales far worse as the image size increases.

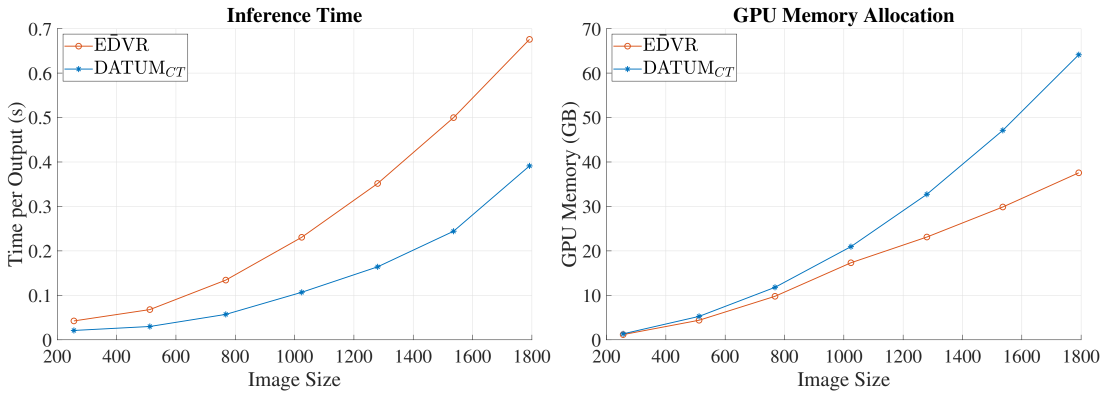


Figure 5.15: Profiling results of E$\overline{\text{D}}$VR and DATUM$_{CT}$ for increasing images size. For all results, the number of input frames was fixed to 15, and for DATUM$_{CT}$ the value of ff/pf was set to 2.

As well as the input image size, the impact of the number of frames used was also be examined, the results of which are shown in Figure 5.16.

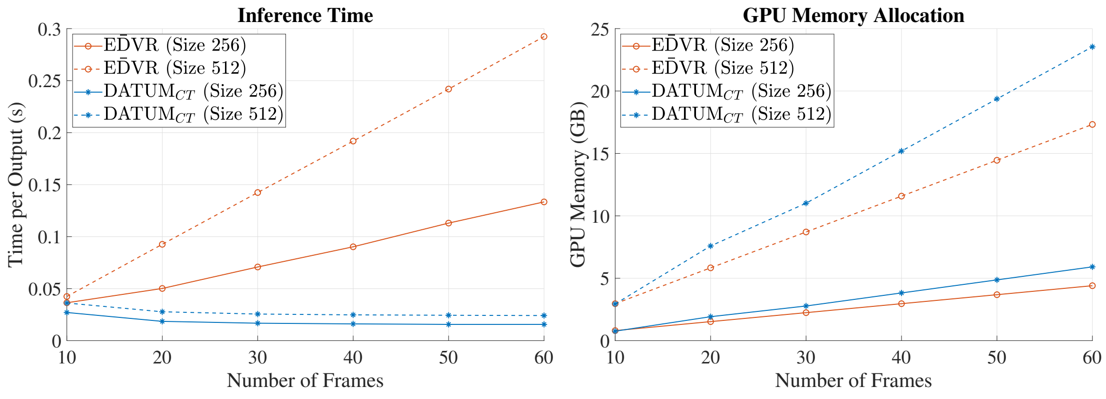


Figure 5.16: Profiling results of E$\overline{\text{D}}$VR and DATUM$_{CT}$ for increasing frame counts. Shown are results for two image sizes. For DATUM$_{CT}$ the value of ff/pf was set to 2.

In these results, the same pattern can be found as before, with E$\overline{\text{D}}$VR requiring more time-per-output than DATUM$_{CT}$, but requiring less memory. The most noticeable difference in these results is the time-per-output of DATUM$_{CT}$, where it can be seen that as more frames are fed into the model, the output frames are actually produced at a faster rate. This is due to the fact that the DATUM$_{CT}$ model processes the whole input video at once and outputs multiple frames (as opposed to E$\overline{\text{D}}$VR, which only outputs a single frame). Therefore, when processing longer video sequences, the total inference time does indeed increase, but

Table 5.23: Profiling results for altering ff/pf values. In all scenarios, the number of input frames was fixed to 15. The profiling of E$\overline{\text{D}}$VR is also shown for reference.

	$\boldsymbol{ff/pf}$	**Frames Out**	**Parameters**	**256 × 256**		**512 × 512**	
				Time	**Memory**	**Time**	**Memory**
DATUM$_{CT}$	0	15	4,486,433	15ms	1.55 GB	22ms	6.14 GB
	2	11	5,779,873	21ms	1.34 GB	30ms	5.26 GB
	7	1	14,389,473	216ms	0.63 GB	300ms	2.34 GB
E$\overline{\text{D}}$VR	N/A	1	23,523,331	42ms	1.17 GB	68ms	4.40 GB

$\boldsymbol{ff}$: Future Frames $\boldsymbol{pf}$: Past Frames

so does the number of output frames, therefore resulting in the time-per-output remaining stable.

The reason for this stability is the manner in which DATUM$_{CT}$ processes a video sequence where, up until the MTCSA module, each frame is processed in the same way. It is here that temporal fusion, the process of fusing a series of frames into one clean image, is applied. By default, DATUM$_{CT}$ fuses together groups of 5 frames at a time (2 future frames, 2 past frames and the central frame, i.e. $ff/pf = 2$). This is the major difference between the two models. Whilst DATUM$_{CT}$ fuses 5 frames at time, E$\overline{\text{D}}$VR fuses the entire input sequence. This therefore allows more refined control of the DATUM$_{CT}$ architecture, as both the number of input frames as well as the value of ff/pf can be chosen. The profiling impact of the ff/pf parameter is shown in Table 5.23.

As expected, the increase of ff/pf also increases the number of parameters within the model, as it is a fundamental change to the architecture. The time-per-output also increases dramatically. This is due to the fact that the entire video sequence is still being processed as before, but now only one output image is being generated, therefore revealing the true inference time of DATUM$_{CT}$. In the case of $ff/pf = 7$, the input/output characteristics of DATUM$_{CT}$ become equivalent to that of E$\overline{\text{D}}$VR (15 frames in, 1 frame out). The results show that, in this case, the E$\overline{\text{D}}$VR model is significantly faster at inference.

Another interesting result is the allocated memory, which decreases as ff/pf increases. This is due to the fact that, in the case of $ff/pf = 7$, the MTCSA only performs a single fusion operation, on all 15 input frames. When $ff/pf = 2$ however, the MTCSA operates on a 'sliding window' of 5 frames. If applied to Eq. (5.5), the indices therefore become

$$(e_{t-2}, e_{t-1}, e_t, e_{t+1}, e_{t+2}), \quad t \in \{2, 3, \ldots, 13\}$$

where t represents each output frame. Therefore, the values of $t = 2$ and $t = 3$ for example, there would exist an 'overlap' where copies of the same frames are created, leading to more total memory allocation during inference.

One element that these scaling studies do not take into account is the turbulence mitigation performance. Whilst the knowledge of inference time and memory allocation is extremely useful, the performance of the configuration should be the first priority. For example, whilst the $ff/pf = 7$ configuration of DATUM$_{CT}$ results in a far longer inference time, it could be the case that this also corresponds to the best performing model. It is therefore worth noting that any comparisons outside of those from Table 5.21 are purely from a hardware deployment point of view, but worthwhile nonetheless.

This comparison between the two models highlights several key factors. First, concerning memory allocation, whilst E$\overline{\text{D}}$VR contained more learnable parameters, it consistently required less memory during inference, allowing easier deployment alongside flexibility of the input size. For example, during testing, the largest image size that was possible for DATUM$_{CT}$ was 1792×1792, whereas E$\overline{\text{D}}$VR could process images up to 2048×2048. As for frame count, with a ff/pf value of 2, DATUM$_{CT}$ maxed out at 700 frames for 256×256 and 200 frames for 512×512. E$\overline{\text{D}}$VR on the other hand managed up to 1700 and 500 frames for 256×256 and 512×512 respectively.

The other key investigation was that of inference time, where it was shown that, per output image, DATUM$_{CT}$ was the faster of the two models. However, if taking into account the full inference time of each model, the DATUM$_{CT}$ architecture takes significantly longer to process all input frames. If such models were being used to perform real-time turbulence mitigation, the DATUM$_{CT}$ model would produce its output in 'chunks', whilst E$\overline{\text{D}}$VR would be able to opperate on a sliding window of the input frames.

As an example, say that turbulence mitigation was required on a 10fps video sequence and the input to each model was set as 20 frames of size 256×256 (Where $ff/pf = 2$ for DATUM$_{CT}$). Given the first 20 frames, DATUM$_{CT}$ would produce 16 output frames, and take approx 304ms. During this time 3 new frames will have been made available. At this point, the 3 new frames could be fed into the model, alongside the most recent 17 frames from the previous batch, however this would lead to duplicate output frames. Another option would be to wait for another 20 frames to be available, which would result in an output of 16 frames every 2s. E$\overline{\text{D}}$VR on the other hand would be able to process the first 20 frames in only 50ms. Once the next frame is then available, the oldest frame can be discarded and a new batch of 20 can be fed in for processing. This therefore is more conducive towards a real-time processing framework, whilst obtaining comparable turbulence mitigation quality as reported in Table 5.21. DATUM$_{CT}$ could be configured and used in the same manner as E$\overline{D}$VR (i.e. set $ff/pf = 7$), however as shown in Table 5.23, each inference would still take 216ms.

This therefore highlights the strength of DATUM$_{CT}$ as a multi-input/multi-output model, rather than a real-time applied model. By using the default value of 2 for ff/pf, the time-per-output frame remains efficient, whilst maintaining model performance. The remaining GPU memory can then be used for as many input frames as possible, as it was shown in the original work [88] as well as in Table 5.19 that the use of more frames leads to a better overall performance. Moreover, the use of more frames allows the model to generate temporally correlated outputs. The fact that the whole video sequences is processed at once, allows the architecture to create an output video that is also consistent between frames. E$\overline{D}$VR on the other hand does not have any knowledge of its previous output frame, so when processing a video using a sliding window, the resulting output video could contain temporal anomalies. This is especially important when processing 'dynamic' data, in which objects may move. DATUM$_{CT}$'s ability to capture temporal information across an entire video will mean that any movement in the sequence is dealt with appropriately, as shown in the original work [88].

A final difference between the two architectures concerns the training process. As described, the original DATUM model, DATUM$_{CT}$, requires an additional 'tilt-free' image to be available during training, whereas E$\overline{D}$VR does not. This would therefore provide more motivation to make use of the E$\overline{D}$VR model, as it allows for more flexibility when choosing training data. However, it has been shown in this chapter that the 'tilt-free' image loss may not be required in all scenarios, so therefore can be removed, allowing the training of DATUM on any dataset.

In summary, the difference between the two models, E$\overline{D}$VR and DATUM$_{CT}$, is not as simple as identifying the model with the best metrics, as both models offer advantages in different scenarios. It was found that, on the whole, the DATUM$_{CT}$ model was the better performing model for turbulence mitigation, however requires more time to process its input fully. This would therefore be put to better use in scenarios where video sequences have already been obtained, and post-processing is required. The DATUM$_{CT}$ model will provide a temporally correlated output video with greatly reduced turblence distortion. E$\overline{D}$VR, whilst not able to fully outperform DATUM$_{CT}$, was however found to be competitive. The manner in which it processes its input is potentially more useful in real-time processing scenarios, where its fast processing time and lower GPU usage may allow a video sequence to be processed frame-by frame, instead of in large ‘chunks’. The two models could therefore be used in tandem, with E$\overline{\text{D}}$VR providing the more deployable real-time solution that could be used for initial evaluation of the video sequence. Once fully obtained, that sequence can then be fed in to DATUM$_{CT}$ to gain the best possible output video.

5.6 Discussion

The results reported in this chapter clearly show that deep learning is well suited for the field of turbulence mitigation. Models are able to understand the underlying distortion, and apply that understanding to unknown samples, providing turbulence mitigated data. As well as this, the models are able to process data in real-time, therefore being a vital tool in any image processing toolbox.

The first key takeaway from the results is the confirmation that video sequences facilitate significantly better turbulence mitigation. Whilst the single frame models were able to provide *some* promising results, the addition of temporally variant information allows the deep learning architectures to better understand the latent, turbulence free image.

The subsequent results from EDVR, as well as the single frame results, also highlight the versatility of deep learning architectures. In both cases, the architectures were not originally designed with turbulence in mind, but were able to adapt after retraining, therefore demonstrating that achieving strong results in deep learning relies more on the dataset, and less on the architectures themselves.

This was also shown by the replacement of the deformable convolution layers with a far simpler design. Not only did this allow an easier training process, but did not result in a loss of performance.

What plays a far more pivotal role in deep learning success is the loss function. If the loss function is not fit for purpose, then the resultant model will fall short of its potential. This was shown constantly in this chapter. By highlighting the fact that turbulence mitigation is primarily an initial processing step within a larger system, attention was brought to the definition of ‘success’ in turbulence mitigation. If, after processing, classification is to be performed, then the training of the model should be undertaken with that in mind.

By using classification as a metric, the outputs of E$\overline{\text{D}}$VR$_{s25}$ and E$\overline{\text{D}}$VR$_{s50}$

were shown to be of better quality, by their higher classification accuracy scores. This therefore indicated that the addition of a perceptual loss into the training process better prepared the model for application based quality metrics. This understanding was then applied to a state-of-the-art model, DATUM, where it was also found that the addition of a perceptual loss resulted in improved performance. Not only this, but it was also found that the tilt loss used in the original work was potentially unnecessary, as the model was able to provide improved results when removed. The removal of this loss also allows this model to be trained on different datasets, as the 'tilt-only' image is no longer required.

Finally, it was shown that, whilst the more modern, turbulence specific, model DATUM$_{CT}$ was able to outperform E$\overline{\text{D}}$VR, in terms real-time applications, the EDVR model would actually be more suitable. This due to how the two models process their data, as well as E$\overline{\text{D}}$VR requiring less memory for inference. The choice between the two simply comes down to the application at hand. If performing real-time processing, E$\overline{\text{D}}$VR may not provide as high quality images, but will be able to process each new image and be deployed on smaller hardware. On the other hand, if a sequence is already obtained, then DATUM$_{CT}$ would be able to process large sections and provide a high quality output video.

5.7 Conclusion

This chapter has presented an in-depth analysis of the use of deep learning for turbulence mitigation, and has presented key findings in its application. It was first found that simple off-the-shelf architectures could be repurposed for single frame turbulence mitigation, and provide somewhat positive results. However it was noted that such a task, given only a single image, was extremely difficult, especially in cases of extreme turbulence. This, alongside inspiration from traditional techniques, therefore motivated the use of turbulent video sequences, providing the the models with a better chance of finding a turbulent free latent image.

This was achieved, again with an off-the-shelf architecture, with EDVR. However, it was found that the training process could be improved by replacing its deformable convolution with a more simple design. As well as this, the loss was altered such that the training process was better suited to turbulence mitigation. These alterations were successful, with the replacement of the deformable convolution resulting in no drop in performance. The addition of a perceptual loss greatly increased the performance of the model, providing images that were visually sharper, and were easier to classify. These findings were then translated over to a more modern, state-of-the-art, turbulence mitigation model DATUM. In this case, it was also found that improvements could be made by altering the loss function, where the perceptual loss, again facilitated better classification, as well as the removal of the tilt loss. This not only provided better results, but also opened up the training of the model to more varied datasets in the future.

Finally, a comparison between the two models was provided. It was shown that the DATUM model was the better performing of the two models, and allowed for more flexibility during testing. However, EDVR was found to be competitive, and more deployable in a real-time application.

In conclusion, the contribution of this chapter is a thorough analysis of turbulence mitigation techniques using deep learning as well as an improvement upon state-of-the-art by optimising the loss function for the training of DATUM.

Chapter 6

Analysis of Image Quality Metrics For Turbulence Mitigation

6.1 Introduction

For every algorithm designed, there must also exist an assurance of quality. In the case of image processing, such a promise of quality can be obtained by employing of a variety of different image quality metrics. These metrics mainly come in two forms: Full Reference (FR) and No Reference (NR). Full Reference metrics utilise the information of a ground truth image, from which it can determine how close an algorithm is to reproducing that ground truth. Conversely, No Reference metrics do not have access to such information, so aim to determine the quality of an image from just the available pixel values alone. The most common metrics used for image processing/recovery algorithms are Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) [27, 165], and are ubiquitous in the field of turbulence mitigation [75, 143, 150, 246].

As shown by the results in Chapter 5, models trained with a perceptual loss are able to outperform others in the task of image classification, even when metrics such as PSNR and SSIM report poorer quality. This is an issue as, if trusted, such metrics would have suggested a model that would have been insufficient for the task at hand. Furthermore, as proposed by Gu et al. [165], deep learning networks may introduce additional texture-like noise in images. This occurs when a model, having been rewarded during training for adding high-frequency detail, begins to apply such detail to unseen data. This results in images that are potentially high quality (due to high frequency content), but also far away from the ground truth (due to noise). It is therefore important to choose an image quality metric that can account for such noise, which is a characteristic lacking in traditional metrics such as PSNR and SSIM.

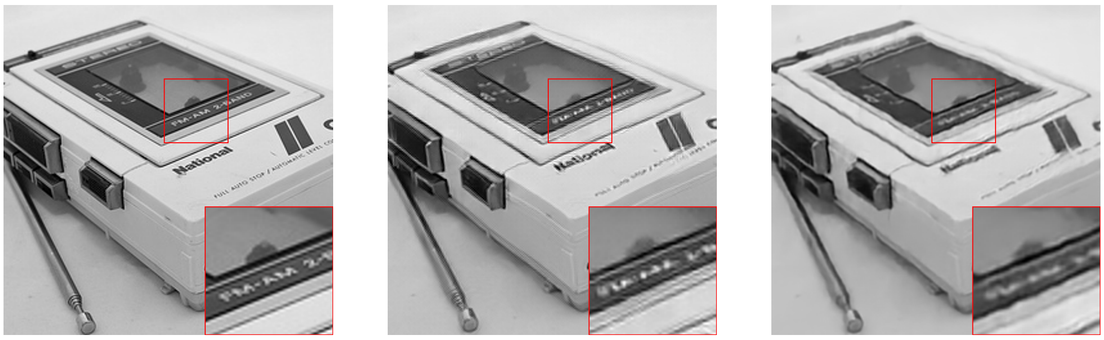
In order to fully understand the potential shortcomings of commonly used metrics, it is first useful to distinguish between perceptual and distortion quality. Distortion quality is focused on the reconstruction of images and is a direct measure of how close an image, $\hat{x}$, is to its reference x. Perceptual quality instead refers to the visual appearance of $\hat{x}$, as opposed to its resemblance to x. An image that looks realistic may have a high perceptual quality, even when very dissimilar to its corresponding reference image. It is therefore common that these two are at odds with each other. As presented in [172], perceptual and distortion quality can be evaluated using NR and FR metrics respectively, with FR metrics typically rewarding images with minimal distortion, while NR metrics aim for perceptual quality.

In the context of turbulence mitigation, when a ground truth image is unavailable, each potential reconstruction of the scene is as equally plausible as the last. However, when a ground truth image becomes available, all but one of these reconstructions would become 'incorrect' when compared to the ground truth (i.e. their distortion quality could be measured and found to be poor). However, these solutions may still have a high perceptual quality and provide useful information regarding the original image contents.

An example is illustrated in Figure 6.1, showing the outputs of two different turbulence mitigation models (A and B), alongside the ground truth image. It can be seen that the output of model A (6.1b) is of higher perceptual quality than that of model B (6.1c). However, a direct comparison with the ground truth (via PSNR) would suggest the opposite. This is therefore a clear example of human visual perception disagreeing with that of distortion metric measurements.

Given the disparity between certain distortion metrics and human perception, it is argued that the field of turbulence mitigation may be hindering its own progress by relying on metrics that fail to align with perceptual quality. If the literature continues to present their results in the form of PSNR/SSIM, the resulting turbulence mitigation algorithms may be solving the wrong problem [165]. The models reported in Chapter 5 show that, for turbulence mitigation, when perceptual quality is taken into consideration, the resulting performance of the post-processing tasks, such as classification, improve greatly.

However, in many cases, using a post-processing step as a metric to evaluate an algorithm's performance is not feasible, therefore requiring the use of image



(a) Ground Truth **(b)** Model A Output (PSNR=19.82dB) **(c)** Model B Output (PSNR=24.23dB)

Figure 6.1: Two example outputs of turbulence mitigation, where the output of the model B is given a lower PSNR score than model A, despite being of better perceptual quality.

quality metrics. Given this understanding, it would therefore be advantageous to identify a metric that is best suited for turbulence mitigation, and that correlates well with application based tests such as classification accuracy.

Image Quality Assessment (IQA) is an extremely well researched field, and new state-of-the-art (SOA) algorithms (For both NR and FR) are constantly being developed [184, 186]. This chapter presents an analysis of such image quality metrics through the lens of turbulence mitigation and evaluates the applicability of such metrics when application-based tests (such as classification) are unavailable. Whilst previous works have approached this problem [57, 88, 145], the analysis presented in this chapter constitutes a far larger investigation, with more metrics, and comprehensive analysis. Such an analysis will be greatly beneficial to the research community, as it will identify metrics that are most suitable for turbulence mitigation quality assessment and, in turn, greatly improve the development future algorithms.

The metrics for this analysis were chosen such that a diverse area of techniques were represented. Modern methods were chosen to assess the state-of-the-art, and more traditional methods were chosen to analyse the performance of these metrics that are ubiquitous to image processing research. The list of metrics, and their descriptions were presented in Chapter 3 (Table 3.2).

The remainder of the chapter is organised as follows. Section 6.2 first compares the chosen metrics against human perception, in order to provide further analysis into the disparity between perceptual and distortion quality. The metrics are then tested against eight different types of distortions over five datasets in Section 6.3 to gain a better understanding of how each metric reacts to certain image features. Finally in Sections 6.4, the metrics are evaluated against outputs of turbulence mitigation models, where each metric is inevitably compared to the classification accuracy of both text data and natural scenes, therefore providing a suggestion as to which metric is most trustworthy in each scenario. Section 6.5 provides the results to each of these studies in turn. Sections 6.6 and 6.7 then discuss and conclude the chapter.

6.2 Turbulence Mitigated Image Quality Survey

In Section 6.1 it was proposed that, of the two images shown in Figure 6.1, one output was of 'higher quality' than the other. However, this was only a single example, and does not guarantee that traditional metrics such as PSNR and SSIM will always disagree with human perception. In order to therefore ascertain if the example shown is a common occurrence, a survey was conducted to gain a wider opinion. To supply images for this survey, two of the models from Chapter 5 were used, notated as Model A and Model B, where the difference between the models was in the loss function. Model A was trained using its default Charbonnier loss function, whereas model B was trained with an additional perceptual loss. As shown in Chapter 5 the inclusion of a perceptual loss results in an improved classification accuracy. It was argued that the reasoning for this improvement was the ability of these models to generate sharper, less distorted images, therefore making the subsequent classification task easier. By utilizing the outputs of these two turbulence mitigation models for this survey, the IQA metrics from Table 3.2 can be evaluated to determine which best align with human perception.

The dataset used for this survey was a subset of the 'turbulent imagenet' dataset presented in Chapter 4. Of the 34,467 original videos, a subset of 5,000 was randomly selected.

The user interface for the survey can be seen in Figure 6.2. The user can toggle between the two versions of the image, where the labelling of 'Version A' and 'Version B' to each version is not tied to the source models, and is randomised for each image to avoid the introduction of bias. Once the user has selected which version 'looks better', they submit their response, and are shown a new image. Survey participants were encouraged to take their time on each image, and skip if they were uncertain. Although 5,000 images were available, participants were told that they could go through as many as they would like. The images shown to each participant were a randomised order of the original 5,000.

Unlike other in depth image quality surveys [165, 192] the purpose of this survey was not to obtain a 'score' of image quality. The purpose was simply to

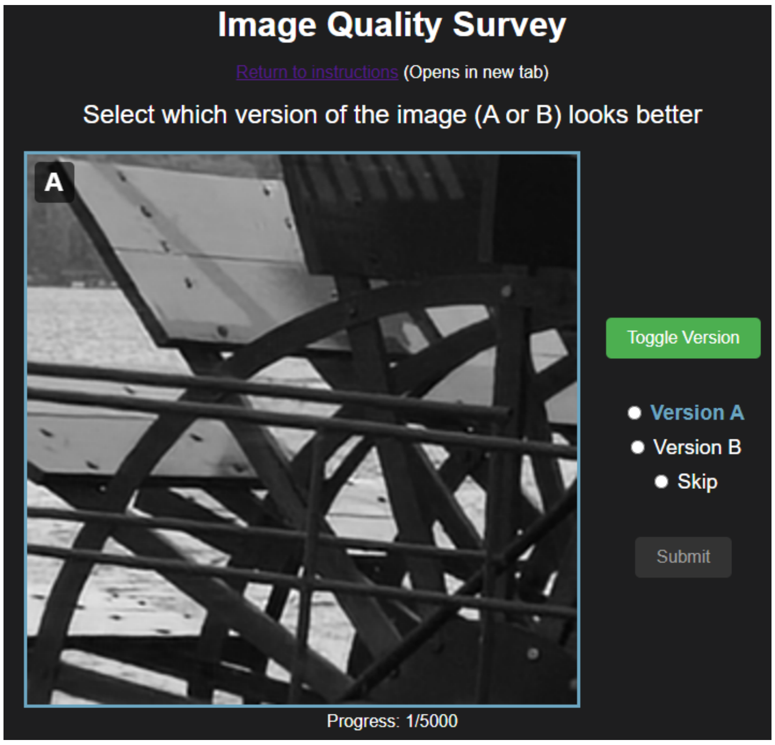


Image Quality Survey
Return to instructions (Opens in new tab)
Select which version of the image (A or B) looks better
A
Toggle Version
Version A
Version B
Skip
Submit
Progress: 1/5000

Figure 6.2: User Interface for Image Quality Survey. Participants were asked to select the version of the image they perceived as 'better quality'.

highlight the disparity between visual perception and the commonly used metrics in research.

6.3 Metric Analysis for Common Image Distortions

Before applying the chosen metrics to turbulence mitigated output images, it is first desirable to understand how each reacts to more traditional distortions. This may allow a better understanding as to why certain metrics perform well or poorly in different scenarios. In order to achieve this understanding, a similar, smaller scale, study as found in [165] was conducted.

The data used for this study originated from 5 separate datasets, with the desire to represent a variety of image contents. These datasets were: Flickr-Faces-HQ (FFHQ) [247], III 5K-word dataset [248], ImageNet [230] and Places [28]. The resulting combined dataset therefore contained contents of faces (FFHQ) , natural scenes (ImageNet & Places) and text (IIIT5K). Also included were two 'Fixed pattern' resolution chart images from the Open Turbulent Image Set (OTIS) [58]. This diverse collection of image content enables the evaluation of each metric's

Table 6.1: Count of images used in metric investigation from each original dataset.

Dataset	Images Used
FFHQ	250
IIIT5K	300
ImageNet	500
Places	500
OTIS	2
Total	1552

applicability to different content types, as well as distortions. In fact, a metric may correlate positively with increased distortion in natural images, but not within text content. The image count taken from each dataset can be seen in Table 6.1.

Alongside the variety in image contents, eight different distortions were chosen; for each distortion there were 10 levels of 'distortion strength', with the exception of turbulence which was represented by 15 levels. The reason for this additional representation was the fact that turbulence was the key motivation for the study, and that the improved fidelity provided by more data would allow a greater understanding of the metric responses. For each distortion type, the lower and upper bounds of "low" and "high" distortion were established through visual analysis of the resulting images, ensuring that the selected limits produced visibly minimal and high degradation, respectively. Intermediate distortion levels were then generated by linearly interpolating between these two extremes. The chosen distortions can be seen in Table 6.2, as well as the units that represent their minimum and maximum levels of distortion. A brief description of how each distortion is applied is provided.

1) Circular Masked Blur is the application of a gaussian blur kernel to a restricted region of the image, where the levels of the distortion relate to the radius of the circular mask used. The MATLAB 'fspecial' function is used to create a gaussian blur kernel of size of 30x30 with a sigma value of 7. It is then applied to the regions of the image outwith the current mask radius. This distortion was included to identify if the metrics would be able to identify region specific image quality. This is useful, as there may be cases where the foregraound is in focus, but the background is blurry. A scenario that would be considered as good quality.

2) Compression is the application of JPEG compression to the original images. By using MATLAB 'imwrite' function, the level of jpeg compression can be provided as a level of quality between 0 (Full compression) and 100 (no compression). This distortion was included to determine whether any of the metrics are sensitive to compression-related artifacts, such as blocking and ringing, which differ fundamentally from blur or noise. This is useful because real-world imaging systems often involve compression, and an ideal metric would ideally penalise overly compressed images.

3) Contrast is the remapping of the maximum and minimum pixel values within the image. The highest contrast is achieved by remapping all pixel values between 0.3 and 0.7 to between 0 and 1. The lowest contrast is represented by remapping the pixels values between 0.1 and 0.9 to 0 and 1. This distortion was included to test whether metrics respond appropriately to changes in global luminance distribution that do not alter spatial structure. This is useful because images with poor contrast can appear washed out or overly harsh, and a good metric should reflect these perceptual effects.

4) Gaussian Noise is simply the addition of noise to the original input. Achieved with MATLAB 'imnoise' function, kernels were generated with a zero mean, and the variance was varied to achieve a range from low noise (0.001) to high noise (0.03). This distortion was included to evaluate how metrics react to random high-frequency artefacts that degrade fine detail. This is useful, as noise is common in many imaging systems, and metrics should be able to distinguish clean images from noisy ones.

5) Motion Blur is most common in handheld photography, as it can be difficult to remain steady during image/video capture. This distortion also makes use of the MATLAB 'fspecial' function, where the differing levels of blur are defined by the 'length' of motion introduced (in pixels). This ranged from low motion (1) to high motion (15). This distortion was included to assess metric sensitivity to directional blur, which differs from isotropic blur both perceptually and structurally. This is useful because motion blur is prevalent in real-world imagery, and metrics should recognise it as a significant degradation of image sharpness.

6) Resizing is the simple act of downsizing the original image by a factor, and then resizing back to the original dimensions. The range of resize factors used for this distortion were between 0.05 (worst) and 0.4 (best). This distortion was included to test whether metrics can detect resolution loss and upsampling artefacts resulting from downsampling. This is useful because many modern imaging tasks—such as super-resolution and compression depend on preserving or restoring fine details lost during resizing.

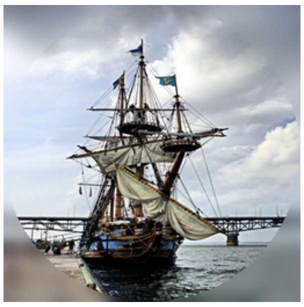
Table 6.2: Values used to impart the minimum and maximum of each distortion type. Where for each distortion, 10 levels of strength were used (*15 levels of turbulence). The manner in which each distortion was applied is also provided.

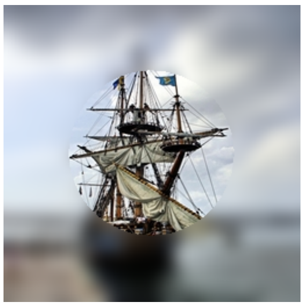
Distortion Type	Minimum Distortion	Maximum Distortion	Units detail
Circular Masked Blur	1	0.2	Diameter of circular mask (As a ratio of image width)
Compression	40	0	jpeg 'imwrite' quality
Contrast	[0.1,0.9]	[0.3,0.7]	Range of values to be remapped to [0,1]
Gaussian Noise	0.001	0.03	Gaussian Noise Variance
Motion Blur	1	15	Pixel Movement
Resizing	0.4	0.05	Downsize ratio
Saturation	1	1 ± 0.5	Saturation Channel Multiplier
Turbulence*	$0.05e^{-15}m^{-2/3}$	$2.5e^{-15}m^{-2/3}$	Cn2 value for simulator

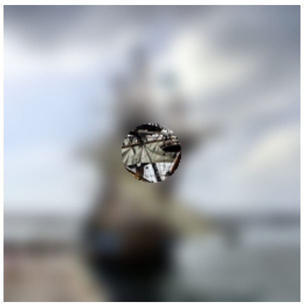
7) Saturation was only applied to the datasets that contained colour images, as saturation is a colour shift operation. Therefore the OTIS dataset was unsuitable. In order to apply saturation shifts, the images first were converted into the Hue, Saturation, value (HSV) colour space with the MATLAB 'rgb2hsv' function. From this, the Saturation channel can be isolated and multiplied by differing factors. This, once restored back to the rgb colour space, results in saturation shifted images. For this distortion, the values of saturation are centred around 1, with a minimum value of 0.5 (Low saturation) and a max value of 1.5 (High saturation). This distortion was included to determine whether metrics are responsive to colour-based degradations, which do not alter image structure or sharpness. This is useful because some metrics focus solely on luminance information, and saturation shifts test their ability, or inability, to account for colour fidelity.

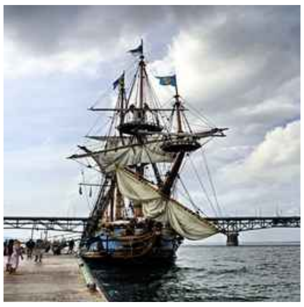
8) Turbulence distortion was generated using the simulator developed in Chapter 4 and was applied in the same manner, using the same settings as the 'turbulence videos' and 'turbulent imagenet' datasets. Unlike those datasets, a wider range of C_n^2 was used, where it varied from $0.05e^{-15}m^{-2/3}$ to $2.5e^{-15}m^{-2/3}$. For the purpose of this investigation, turbulence distortion was only applied to the Places and ImageNet images within the dataset. As the true goal of this section is to identify a suitable metric for turbulence mitigation, the first step is to determine which metrics can reliably distinguish between different levels of turbulence before processing.

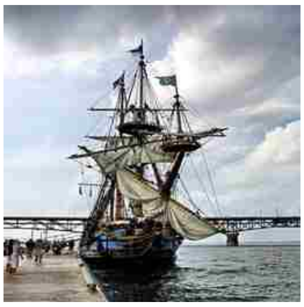
Examples of each distortion are shown in Figures 6.3 and 6.4.

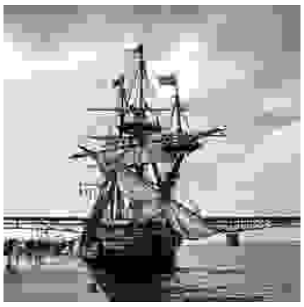

(a) Circular Mask Blur: 1.00

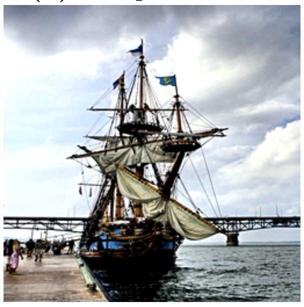

(b) Circular Mask Blur: 0.56

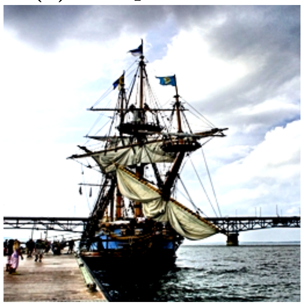

(c) Circular Mask Blur: 0.20

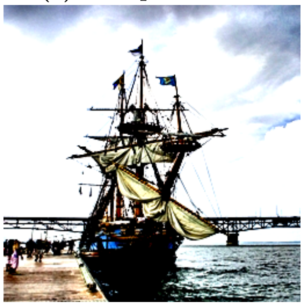

(d) Compression: 40

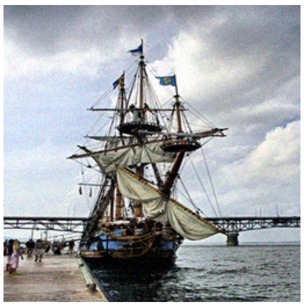

(e) Compression: 18

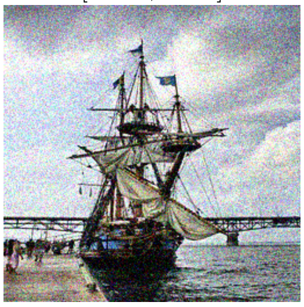

(f) Compression: 0


(g) Contrast: [0.1,0.9]


(h) Contrast: [0.189,0.811]


(i) Contrast: [0.3,0.7]


(j) Gaussian Noise: 0.001


(k) Gaussian Noise: 0.014

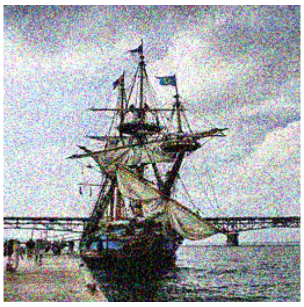
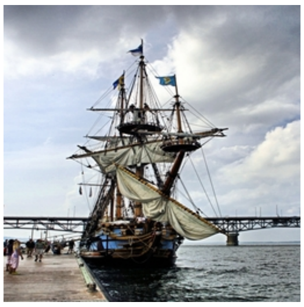
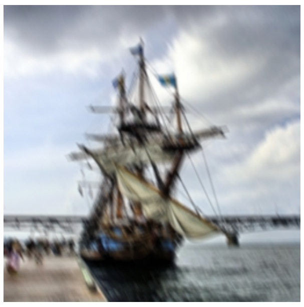
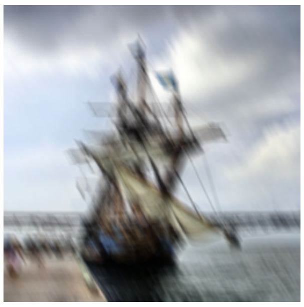
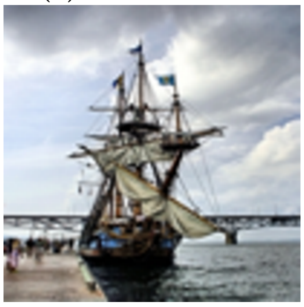

(l) Gaussian Noise: 0.030

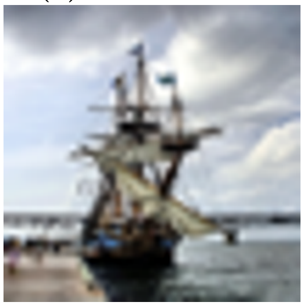
Figure 6.3: Example images of the first four of the chosen distortions: Circular Mask Blur, Compression, Contrast and Gaussian Noise. Shown are examples of the minimum distortion (Left), medium distortion (Middle) and maximum distortion (Right).

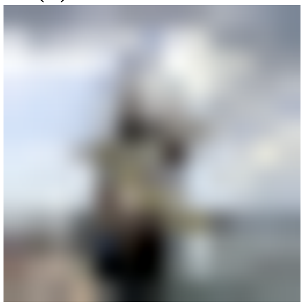

(a) Motion Blur: 1

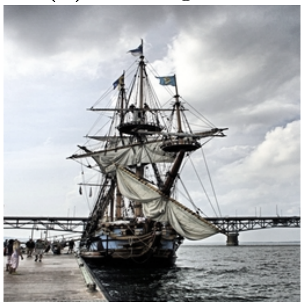

(b) Motion Blur: 7

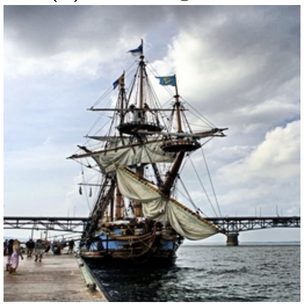

(c) Motion Blur: 15

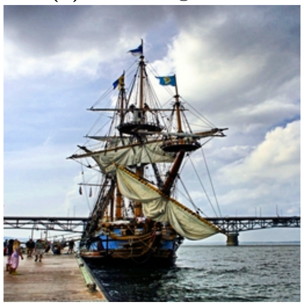

(d) Resizing: 0.40

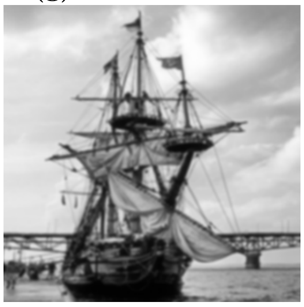

(e) Resizing: 0.21

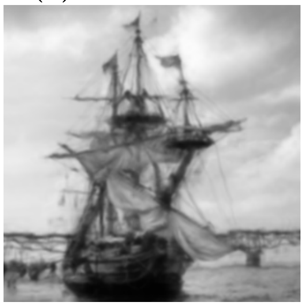

(f) Resizing: 0.05


(g) Saturation: 0.50


(h) Saturation: 0.94


(i) Saturation: 1.50


(j) Turbulence: $0.05e^{-15}m^{-2/3}$


(k) Turbulence: $1.275e^{-15}m^{-2/3}$

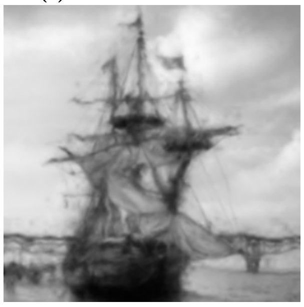

(l) Turbulence: $2.5e^{-15}m^{-2/3}$

Figure 6.4: Example images of the second four of the chosen distortions: Motion Blur, Resizing, Saturation and Turbulence. Shown are examples of the minimum distortion (Left), medium distortion (Middle) and maximum distortion (Right). With the exception of Saturation, which is displayed as: most Desaturated (Left), Normal (Middle) and most Saturated (Right).

Table 6.3: Total image count for each combination of distortion and dataset, where the count of available images per dataset are presented in Table 6.1, and each distortion has 10 levels (15 for turbulence).

Distortion	OTIS	FFHQ	IIIT5K	ImageNet	Places	Total per distortion
Masked Blur	20	2500	3000	5000	5000	15520
Compression	20	2500	3000	5000	5000	15520
Contrast	20	2500	3000	5000	5000	15520
Noise	20	2500	3000	5000	5000	15520
Blur	20	2500	3000	5000	5000	15520
Resizing	20	2500	3000	5000	5000	15520
Saturation	-	2500	3000	5000	5000	15500
Turbulence	-	-	-	7500	7500	15000
					Total	**123,620**

Given 5 original datasets, 8 distortions and 10 levels per distortion (15 for turbulence), the final dataset contained a total of 123,620 images. A more detailed breakdown can be seen in Table 6.3.

Using this dataset, a full analysis of the chosen metrics could be undertaken, where their applicability to different image contents and distortions could be identified. This was as simple as taking each of the 123,620 images in the dataset and obtaining a score from each of the metrics.

In order to form an analysis of the results of such an investigation, the correlation between the original distortion 'strengths' and the metric response can be calculated. For this, the Spearman rank-order correlation (SRCC) and Pearson linear correlation (PLCC) metrics can combined to form a final score, as suggested in the NTIRE 2021 competition [179] (and continued in 2022 [27]). This final score (or 'main score') is evaluated as

$$MainScore = |SRCC| + |PLCC| \tag{6.1}$$

where the PLCC and SRCC scores are evaluated using MATLABs 'corr' function and, unlike previous works [27], are evaluated using all available data samples instead of first evaluating a 3rd order polynomial. Each metric score corresponds to a unique distortion value, therefore a 3rd order polynomial would reduce the data into only 10 points (15 for turbulence), and remove the detail gained by evaluating the correlation over all the data. For the remainder of the chapter, the term 'correlation' is used to describe the quality of a given metric. If a metric is described to have good correlation, then as the image distortion increases, that metric has, in turn, provided a poorer score (where 'poorer' depends on the nature of the metric, either ↑ or ↓).

6.4 Metric Analysis for Turbulence Mitigation

6.4.1 Observed Distortions

The previous study allows for a better understanding of each metric when faced with different data and distortions. This however is not the core investigation of this chapter as the investigation of metrics on distortions has been an ongoing field of study [166, 185, 188, 196, 197]. However, as highlighted by Jinjin et al. [165], the simple assessment of metrics on common distortions is not able to tell a full story when it comes to Image Recovery (IR) algorithms, especially in the modern era of data driven deep neural networks. Jinjin et al. highlight the prevalence of Generative Adversarial Network (GAN) based IR algorithms, and state that while they are easily able to generate realistic images, the resulting images may contain "fake details and textures" [165]. They therefore include such GAN generated images within their dataset for metric analysis. Such a study is yet to be performed for the task of turbulence mitigation (TM), and is therefore presented here. The aim of such a study is to find a metric that correlates best with image quality after processing. For example, Zhang et al. [88] find that in the case of turbulence distortion, the CW-SSIM metric presents the best correlation with increasing turbulence distortion. However, this does not necessarily justify its correlation with the image quality *after* being processed. By the nature of deep learning, the model may impart details from what it has identified during the learning process. The resulting image quality therefore may no longer correlate with previously assumed metrics.

This section aims to perform an initial study of how metrics respond to turbulence mitigated images. During the training processes in Chapter 5, many different models were trained, with varied levels of success. These are utilised in this section to generate images that, to a human perspective, are of differing quality.

Two datasets were used for this generation process: the ATSyn-static [88] test dataset and the Turbulence Text dataset [60].

The ATSyn-static data consists of natural scene images, similar to that which was used to train each of these models. The text dataset simply contains images of text printed onto a piece of paper; not only is this type of image contents different from the training data, but was also obtained without the use of a simulator with the turbulent distortion being captured in camera. For this dataset, the available videos were utilized in two ways: firstly by feeding the model 20 input frames per output frame Text (20), and secondly by feeding the model 60 input frames per output frame Text (60).

For each of these three potential output image types (ATSyn-static, Text (20) & Text (60)), models from Chapter 5 were chosen based on the 'observed distortion' in the output images. Where the observed distortion was from a single source, and was restricted to a single type, i.e. blurry/sharp or noisy/clean, and did not represent the observed *quality* of the images, as it is possible for text to remain highly readable even within a noisy image. The details of the models used for each dataset are reported in Table 6.4.

Table 6.4: Architecture details for each model chosen for each dataset. As well as the observation as to the output image quality. Each model is referred to by their respective 'Reference Name' in the main text.

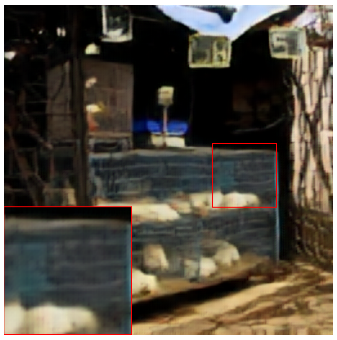
Dataset	**Architecture**	**Sheduler**	**Perceptual Loss**	**Observed Distortion**	**Reference Name**
ATSyn-static	DATUM	Cosine	-	Blurry	Blurred_1
	DATUM$_s$	Plateau	-	Blurry	Blurred_2
	DATUM$_s$	Cosine	Low	Sharp	Sharp_1
	DATUM	Cosine	Medium	Sharp	Sharp_2
	DATUM	Plateau	Medium	Anomaly	Anomaly_ATSyn-static
Text (20)	DATUM	Cosine	-	Low Noise	Low_Noise_20
	E$\overline{\text{D}}$VR	Plateau	-	Med Noise	Med_Noise_20
	E$\overline{\text{D}}$VR$_s$	Plateau	V. High	High Noise	High_Noise_20
Text (60)	DATUM	Cosine	-	Low Noise	Low_Noise_60
	DATUM$_s$	Cosine	Low	Med Noise	Med_Noise_60
	DATUM$_s$	Cosine	High	High Noise	High_Noise_60
	DATUM	Plateau	Medium	Anomaly	Anomaly_Text

As mentioned, these models were originally trained to varied success, meaning that the training procedure between each model is not consistent. This however is not an issue, as it is the relationship between the output images and the metrics that is under investigation, not the quality of models themselves.

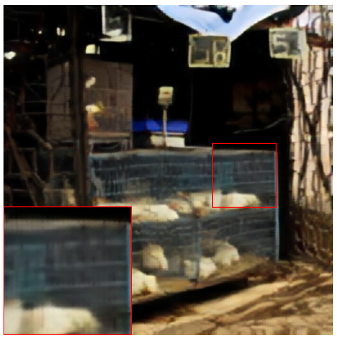
Of the details provided in Table 6.4, E$\overline{\text{D}}$VR represents the original E$\overline{\text{D}}$VR architecture [233] with the deformable convolution replaced as in Chapter 5. E$\overline{\text{D}}$VR$_s$ represents a smaller version, with 64 features and 10 reconstruction blocks (Instead of 128 and 40). DATUM represents the original architecture [88], and DATUM$_s$ again represents a smaller version with only 8 features (instead of 16).

Models that were trained with a cosine scheduler had a fixed number of training iterations, as the learning rate dropped in a cosine manner. The models trained with a plateau scheduler were unbounded, and were allowed to train until the loss dropped too low. Different models also contain different levels of influence from a perceptual loss element, the level of which is classified as Low (Less impact than the default loss), Medium (Slightly more impact than the default loss), High (More impact than the default loss), and V. high (much more impact than the default loss).

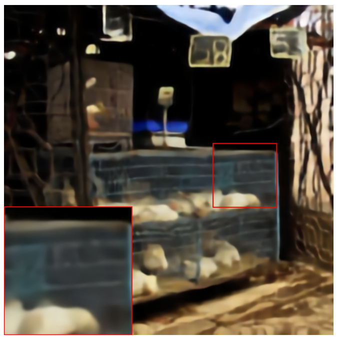
Example output images for each model on their respective datasets are shown in Figures 6.5, 6.6, 6.7 and 6.8.



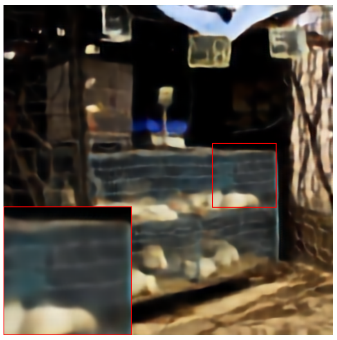
(a) Output from 'Sharp_1' model



(b) Output from 'Sharp_2' model

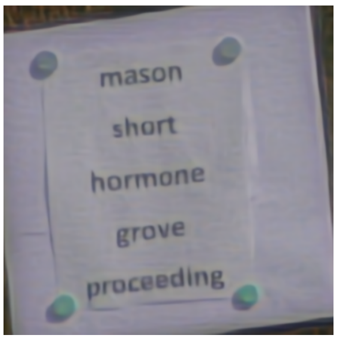


(c) Output from 'Blurred_1' model

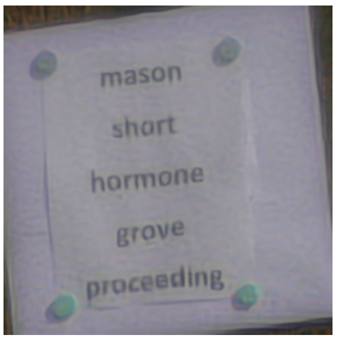


(d) Output from 'Blurred_2' model

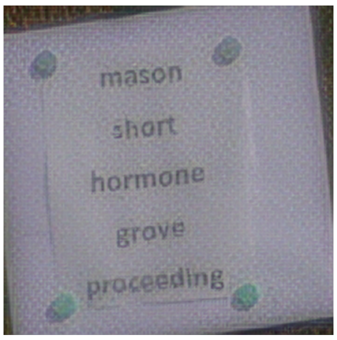
Figure 6.5: Example outputs of the chosen ATSyn-static models. Showing the difference between what was observed as 'Sharp' or 'Blurry'.



(a) Low Noise

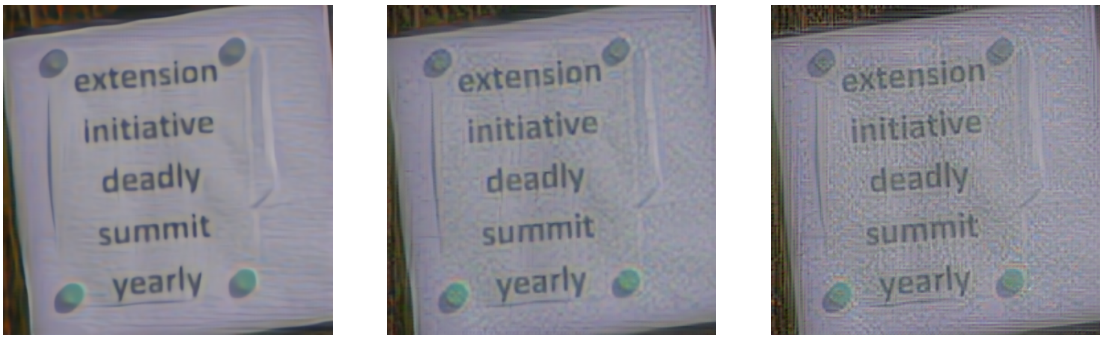


(b) Medium Noise



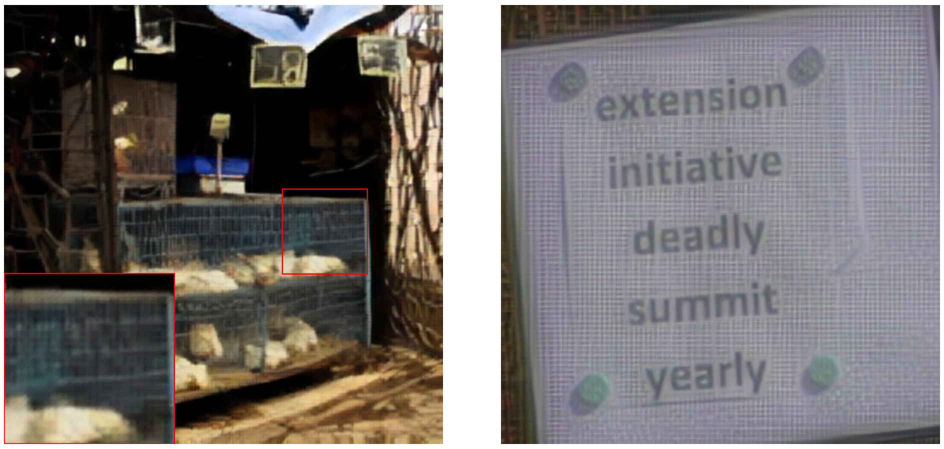
(c) High Noise

Figure 6.6: Example outputs of the Text (20) models, with the observation ranging from Low to High Noise.



(a) Low Noise (b) Medium Noise (c) High Noise

Figure 6.7: Example outputs of the Text (60) models, with the observation ranging from Low to High Noise.



(a) ATSyn-static Anomaly (b) Turbulence Text Dataset (60) Anomaly

Figure 6.8: Example output images from the anomaly model, where the resulting images contain noise in an interesting uniform pattern.

For the ATSyn-static data (Figure 6.5), it is difficult to initially notice the differences between each model output, however once examined more closely, differences become clearer. These observations are classified as either 'blurry' or 'sharp'.

The models were trained on natural scenes, meaning that they learned to include high frequency detail in their outputs. When presented with 'synthetic' images however, this high frequency content manifests as noise, as is the case for the Turbulence Text dataset (Figures 6.6 and 6.7). Therefore, the observations of these outputs are identified as three levels of noise in the images: Low, Medium and High. Finally, a model was chosen due to its ability to impart an 'interesting' distortion to the output images (Figure 6.8). This model has been noted as 'Anomaly' in Table 6.4.

Using the output images from the selected models on their respective datasets, a metric analysis can be performed. This will allow an initial understanding of how each metric reacts to the different levels of observed distortions in the output images, with the aim to find the metric that is able to consistently identify the differences in the quality.

6.4.2 Metrics for Application

When applying image processing algorithms, there must also be a reason for doing so. For example, super resolution is used to upsample old data for a new (larger) format, or segmentation can be used to identify multiple subjects in an image without human supervision. Turbulence mitigation algorithms are most likely used as an initial step in a larger processing chain, where it could be followed by tasks such as detection, classification or tracking. The fundamental goal of turbulence mitigation, therefore, is to assist post-processing algorithms by first improving the image/video quality, and should therefore be trained with such post-processing in mind.

Traditionally, the best metric for a task would be identified by determining its correlation with human scores [165, 185, 188]. Achieved via detailed survey techniques, images are shown to a large population of human observers to obtain a mean opinion score (MOS) for each image. These can then be compared with the metrics under investigation, allowing the identification of the metric that best agrees with human perception. However, this is a lengthy process and not only requires a suitably large number of observers, but also observers that are capable of providing good quality feedback. Furthermore, this process still does not align with the overall goal of post-processing success. Just because a human perceives an image to be of good quality, that does not infer that it is also easily classifiable. This is especially true if the post-processing classification/tracking is an automated task.

With this in mind, it would therefore be desirable to identify metrics based on their ability to maximise the efficiency of post-processing techniques. Not only will this aid in the future development of TM algorithms, but the resulting metric could also be utilised in scenarios where classification/tracking knowledge is unavailable. If a metric can be found that correlates well with classification accuracy, it can then instead be used as a 'prediction of classification' where, by setting a threshold, an image can be labelled as 'classifiable' or 'unclassifiable' depending on the metric score.

To identify if any of the chosen metrics correlate well with classification, two models were chosen: A version of $\text{E}\overline{\text{D}}\text{VR}$ trained with a perceptual loss, and a version of DATUM also trained with a perceptual loss. These models were then used to process two turbulent datasets: the same subset of the ‘turbulent imagenet’ dataset used in the image quality survey (5000 videos) and the Turbulence Text Dataset [60] dataset, the contents of which are able to be processed by a text classification model. The output images were then also subject to metric evaluation, therefore providing a direct comparison between metric scores, and classification accuracy.

As well as the difference between the two models’ architecture, two different datasets were used for the training process, where the ED̄VR model was trained using the ‘turbulence videos’ dataset proposed in Chapter 4 and the DATUM model trained with the ATSyn-static dataset [88]. The choice of the models and their training datasets provided a good range of potential turbulence mitigation success. It was expected for example, that the ED̄VR model trained on ‘turbulence videos’ would perform the best on the ‘turbulent imagenet’ test data, as the two datasets are sourced from the same simulator, therefore giving the ED̄VR model an advantage. The difference in model quality was not to highlight the difference between the models themselves, but to allow a full analysis of the metrics. By having two models with differing ability, the consistency of the metrics can be fully understood.

In order to obtain the classifications of the model outputs, off-the-shelf classifiers were used. In the case of the ‘turbulent imagenet’ data, the EfficientNetV2-L [21] classifier was used. For these results, it is worth highlighting that the ground truth images from the ‘turbulent imagenet’ subset are classifiable with 100% accuracy with this classifier, meaning that any images that are incorrectly classified after turbulence mitigation are so solely due to the distortions present.

For the case of text classification, as in [88], three text classification models were utilised: CRNN [243], DAN [244] and ASTER [245]. The Turbulence Text dataset is supplied with the correct bounding box for each word, where there are five words per image/video. Using these, each word is first cropped out of the image before being processed by the text classification models. Another factor for this dataset is the fact that each of the 100 videos contains 100 frames. The two models utilize frames differently: the ED̄VR model takes 15 frames as input for each output frame, while DATUM adjusts the number of input and output frames as per its configuration. For ED̄VR therefore, the 100 available frames are separated into 6 ‘sub videos’ with 15 frames each, resulting in 6 output frames per video. For DATUM, the model was configured to take 60 frames (from the centre of the 100) as input, and produce 4 output frames. Given these input/output configurations, alongside the fact that each image contains five words, of the 100 input videos, the E$\overline{\text{D}}$VR generates a total of 3,000 output samples, and DATUM generates 2,000.

The application of the metrics to the 'turbulent imagenet' output images was achieved in the same manner as in previous sections, with each metric assessing the quality of each image. For the text data however, the metrics were provided with the whole images, as opposed to the cropped text. This therefore introduces a disparity between what the metric is reporting, and the classification ability of the text. However, the bounding box of text is very rarely available outside carefully labelled datasets, so the application of metrics to the whole image was deemed more appropriate.

Once each of the datasets had been processed by the models, the classifications obtained and the metrics evaluated, the correlation between metrics and classifications could then be found. This was done in the same manner as the distortion study in Section 6.3, where the PLCC and SRCC scores were combined to evaluate the main score for each metric, as per Eq. (6.1).

The final step in this investigation is to then find an optimal threshold for each metric, and evaluate its performance in a scenario where classification is unavailable. To achieve such an understanding for each scenario, the Balanced Accuracy (BAcc) can be calculated for each metric. As detailed in Chapter 3, BAcc is an evaluation metric that aims to balance the True Positive Rate (TPR) and True Negative Rate (TNR). It represents the probability of classification accuracy, where, in an ideal case, the two distributions of classifications would be successfully separated by the threshold, resulting in a BAcc 100% (i.e. the threshold can directly separate TP and TN predictions). In the worst case, the two distributions would be identical, resulting in a BAcc of 50% (i.e. the best threshold would be as good as random choice). Using the optimal threshold τ^* for each metric, each image can be labelled as either 'classifiable' or 'unclassifiable', depending on the metric score.

6.5 Results

6.5.1 Image Quality Survey

The image quality survey had 53 participants, with 1,825 unique image pairs shown and 2,387 total evaluations submitted. Of these evaluations, 2,150 identified model B as the better quality image, while 189 evaluations identified model A. The remaining 48 responses were cases that were skipped by the participant, indicating that there was difficulty in finding a discernable difference between the two images.

With 1,825 unique image pairs and 2,387 evaluations, inevitably some image pairs were evaluated more than once. There were 445 such image pairs, where model B was unanimously chosen for 355, and model A was chosen for only 2. Of the remaining 88 image pairs, neither model A or B received a unanimous vote. The overall distribution of these 445 pairs can be seen in Table 6.5.

Table 6.5: Evaluation distribution for image pairs shown more than once. Displaying the percentage of the evaluations for each option, and the number of times each evaluation split occurred.

Model A	Model B	Skipped	Occurrences
0.0%	0.0%	100.0%	0
0.0%	100.0%	0.0%	355
100.0%	0.0%	0.0%	2
0.0%	66.7%	33.3%	1
0.0%	50.0%	50.0%	7
20.0%	80.0%	0.0%	1
25.0%	75.0%	0.0%	1
25.0%	50.0%	25.0%	1
33.3%	66.7%	0.0%	14
33.3%	33.3%	33.3%	1
40.0%	40.0%	20.0%	1
50.0%	50.0%	0.0%	55
50.0%	0.0%	50.0%	4
66.7%	0.0%	33.3%	1
75.0%	0.0%	25.0%	1

These results therefore align with the findings in Chapter 5, where the addition of perceptual loss to the training procedure results in higher quality output images. That was not the core purpose of this survey; rather, its goal was to identify the metrics that agree or disagree with human perception. To this aim, Table 6.6 shows how each metric scored the outputs from each model.

It can be seen that all of the NR metrics, with the exception of QualiCLIP, agree with the findings of the image quality survey, and identify the outputs from model B as superior. Within the FR metrics, the distinction between the two models is not as clear, with the majority of metrics (7/11) deeming the outputs of model A to be better, therefore disagreeing with the human perception. The remaining metrics, i.e. DISTS, LPIPS, PieAPP and TOPIQ_FR agree with human perception, and are therefore potential FR metrics that could be better

Table 6.6: Metric responses from the 1825 unique survey image pairs from Models A and B, showing that NR metrics agree that the outputs of Model B are of better quality, whereas the FR metrics, on the whole, suggest the opposite.

	Metric	Model A	Model B
NR	ARNIQA ↑	0.631	**0.640**
	BRISQUE ↓	29.903	**22.788**
	CLIP_IQA ↑	0.606	**0.651**
	LIQE ↑	3.374	**3.664**
	MANIQA ↑	0.374	**0.439**
	NIQE ↓	6.038	**5.430**
	PI ↓	4.285	**3.828**
	PIQUE ↓	37.139	**27.271**
	Q-ALIGN ↑	2.304	**2.473**
	QualiCLIP ↑	**0.682**	0.670
	TOPIQ_NR ↑	0.501	**0.590**
FR	AHIQ ↑	**0.511**	0.493
	CW-SSIM ↑	**0.997**	0.993
	DISTS ↓	0.124	**0.074**
	FSIM ↑	**0.909**	0.869
	LPIPS ↓	0.105	**0.066**
	MSSSIM ↑	**0.967**	0.926
	PieAPP ↓	0.474	**0.363**
	PSNR ↑	**26.775**	23.813
	SSIM ↑	**0.829**	0.724
	TOPIQ_FR ↑	0.831	**0.877**

used in the quality assessment of turbulence mitigated images. It is interesting to note that these FR metrics that agree with human perception are all data driven metrics, which is likely the reason for their correlation.

The key conclusion to be made from this study, and from Table 6.6, is that different methods commonly used to evaluate image quality are not wholly in agreement. The results show that the NR metrics tend to agree with human perception, likely due to their nature to assess the image quality via statistical analysis and learnt features. The FR metrics on the other hand tend to disagree with human perception, indicating that they, instead of promoting feature rich, high quality images, care more about raw pixel values and the ability for an algorithm to directly reconstruct the ground truth. This is as expected, as similar studies have reported similar results [69]. This however is the first such study to be performed for the task of turbulence mitigation, and will provide valuable information regarding the metrics to trust when evaluating model performance.

6.5.2 Common Distortions

For a detailed understanding of each metric, and how they respond to image contents and distortions, the analysis was divided into two approaches. The first approach examined each metric within each unique dataset, averaging the results across all distortions. This allows an evaluation of each metric on each data type, regardless of the specific distortions. The second approach takes the opposite perspective by analysing each metric for each type of distortion, averaging the results across all datasets. In this way, the focus is on assessing the impact on the metric by the distortions, independent of the dataset used.

For each of these approaches, the same analysis procedure was followed. Firstly, the correlation between each metric and the datasets/distortions is calculated in the form of a main score, which is a cumulation of the SRCC and PLCC scores (Eq. (6.1)). Once the correlation of each metric is known, they are then ranked such that the metric with the highest correlation is ranked as 1, and the metric with the least correlation ranked 22. The final rank of each metric can then be obtained by summing the ranks over each of the the datasets/distortions. This analysis makes it possible to find the metric that is most likely to be informative no matter the distortion/dataset.

Impact of Dataset

The datasets used in this study included a diverse range of image content, enabling an analysis of how each metric varied with different types of image contents. Figure 6.9 shows the main score of each metric with respect to each dataset, independent of the distortions. After ranking each metric on each dataset, the resulting total ranks of each can be seen in Table 6.7.

The first insight to be gained from this graph is that all metrics correlate positively (i.e. the metrics correctly assign better scores to lower distortions) with the increasing levels of distortion on each dataset (taking into account the polarity of each metric). It can also be seen that the full reference metrics reliably outperform the no reference metrics. For all the datasets except OTIS, the highest performing FR metrics are TOPIQ_FR and DISTS. This can also be observed in the rankings in Table 6.7.

However when assessing the more 'synthetic' contents of the OTIS resolution charts, the metrics with the highest correlations are FSIM and SSIM. This therefore shows that, just because a metric is suitable for one type of image contents, this does not mean such a metric is suitable for all.

A similar pattern can be found within the NR metrics where, for all datasets

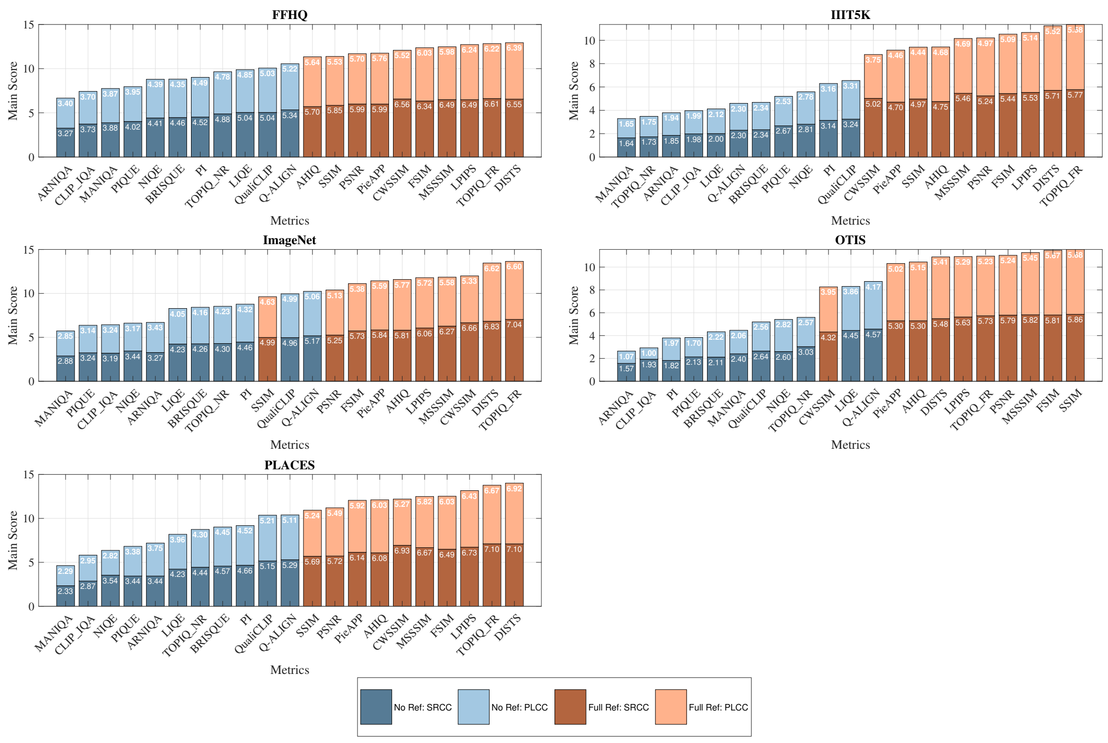


Figure 6.9: Correlation ($Main\ score = PLCC + SRCC$) between each metric and each dataset used (Zoom in for best viewing).

except IIIT5K, the best correlation is found with the Q-ALIGN metric. This then changes to QualiCLIP for the text contents of IIIT5K, with Q-ALIGN ranking 6th of the NR metrics. This suggests that the LLM approach of Q-ALIGN struggles with the content of the text scenes of IIIT5K.

When comparing the performance of NR vs FR, it was expected that the correlation of the FR metrics would outperform that of the NR. This is simply due to the presence of the ground truth, with the knowledge of what each image should look like, the FR metrics are far more capable at noticing the differences. From these results however, it is positive to note that for the more ‘natural’ images of FFHQ, ImageNet and PLACES, the NR metrics are able to provide high correlation scores. In the case of IIIT5K, the difference between NR and FR is more distinct. This is most likely due to the fact that the images in this dataset are of text/patterns which, by their nature will contain edges and sharp contrast changes. It is therefore likely that the difference between each distortion level was not enough to result in noticeable differences on these images, as they would on the more natural scene data. The NR metrics would then have judged each level of distortion similarly, resulting in poor correlation.

Interestingly, the results on the OTIS dataset seem to present as three different ‘levels’ of correlation, with the majority of FR metrics performing well, three

Table 6.7: The individual ranking of each metric on each dataset (Based on main score). The sum (And re-ranking) of these therefore provide an idea of which metric is best suited across **all** datasets (The lower the better).

	Metric	FFHQ	IIIT5K	ImageNet	OTIS	PLACES	Overall Ranking
NR	ARNIQA	21 (11)	19 (9)	17 (7)	21 (11)	17 (7)	19 (9)
	BRISQUE	16 (6)	15 (5)	15 (5)	17 (7)	14 (4)	16 (6)
	CLIP_IQA	20 (10)	18 (8)	19 (9)	20 (10)	20 (10)	20 (10)
	LIQE	13 (3)	17 (7)	16 (6)	11 (2)	16 (6)	14 (4)
	MANIQA	19 (9)	21 (11)	21 (11)	16 (6)	21 (11)	21 (11)
	NIQE	17 (7)	13 (3)	18 (8)	14 (4)	19 (9)	17 (7)
	PI	15 (5)	12 (2)	13 (3)	19 (9)	13 (3)	13 (3)
	PIQUE	18 (8)	14 (4)	20 (10)	18 (8)	18 (8)	18 (8)
	Q-ALIGN	11 (1)	16 (6)	10 (1)	10 (1)	11 (1)	11 (1)
	QualiCLIP	12 (2)	11 (1)	11 (2)	15 (5)	12 (2)	12 (2)
	TOPIQ_NR	14 (4)	20 (10)	14 (4)	13 (3)	15 (5)	15 (5)
FR	AHIQ	10 (10)	7 (7)	6 (6)	8 (8)	7 (7)	8 (7)
	CW-SSIM	6 (6)	10 (10)	3 (3)	12 (10)	6 (6)	7 (6)
	DISTS	1 (1)	2 (2)	2 (2)	7 (7)	1 (1)	2 (2)
	FSIM	5 (5)	4 (4)	8 (8)	2 (2)	4 (4)	5 (5)
	LPIPS	3 (3)	3 (3)	5 (5)	6 (6)	3 (3)	3 (3)
	MSSSIM	4 (4)	6 (6)	4 (4)	3 (3)	5 (5)	4 (4)
	PieAPP	7 (7)	9 (9)	7 (7)	9 (9)	8 (8)	9 (7)
	PSNR	8 (8)	5 (5)	9 (9)	4 (4)	9 (9)	6 (6)
	SSIM	9 (9)	8 (8)	12 (10)	1 (1)	10 (10)	10 (7)
	TOPIQ_FR	2 (2)	1 (1)	1 (1)	5 (5)	2 (2)	1 (1)

.

metrics showing moderate performance, and the remaining NR metrics performing poorly. These show that while, on the whole, the NR metrics are less useful than FR in the case of high contrast pattern images, the Q-ALIGN and LIQE metrics are able to significantly outperform the other NR competition.

As with the results from the image quality survey, it can be seen that, of the FR metrics, it is the data driven methodologies that provide the greatest correlation (TOPIQ_FR, DISTS & LPIPS). This again shows that modern deep learning techniques are able to outperform their more traditional counterparts, it is therefore likely that this improved correlation is due to their ability to learn from real, suitable data. Another architectural insight to these models is the shared technique of evaluating image quality at multiple scales. This technique clearly allows each to gain a fuller understanding of the image contents. As for NR metrics, the Q-ALIGN metric is the only to employ the use of a large language model, which in turn makes use of a vision transformer. The improved capability of this metric shows that whilst a metric such as MANIQA also employs a vision transformer, the use of an LLM allows Q-ALIGN to gain more ‘freedom’ with its quality assessment, therefore resulting in higher correlations.

Impact of Distortion

To gain a full understanding of the capability of each metric, the analysis of each for each distortion is critical. Any metrics that would be able to provide consistent results over multiple modalities would be extremely beneficial. By averaging over all of the datasets for each distortion, any bias from data contents is removed, and the metric response is due purely to the distortion type, independent of image contents. Figure 6.10 shows the resulting main scores of each metric for each distortion, with Table 6.8 showing the resulting rankings.

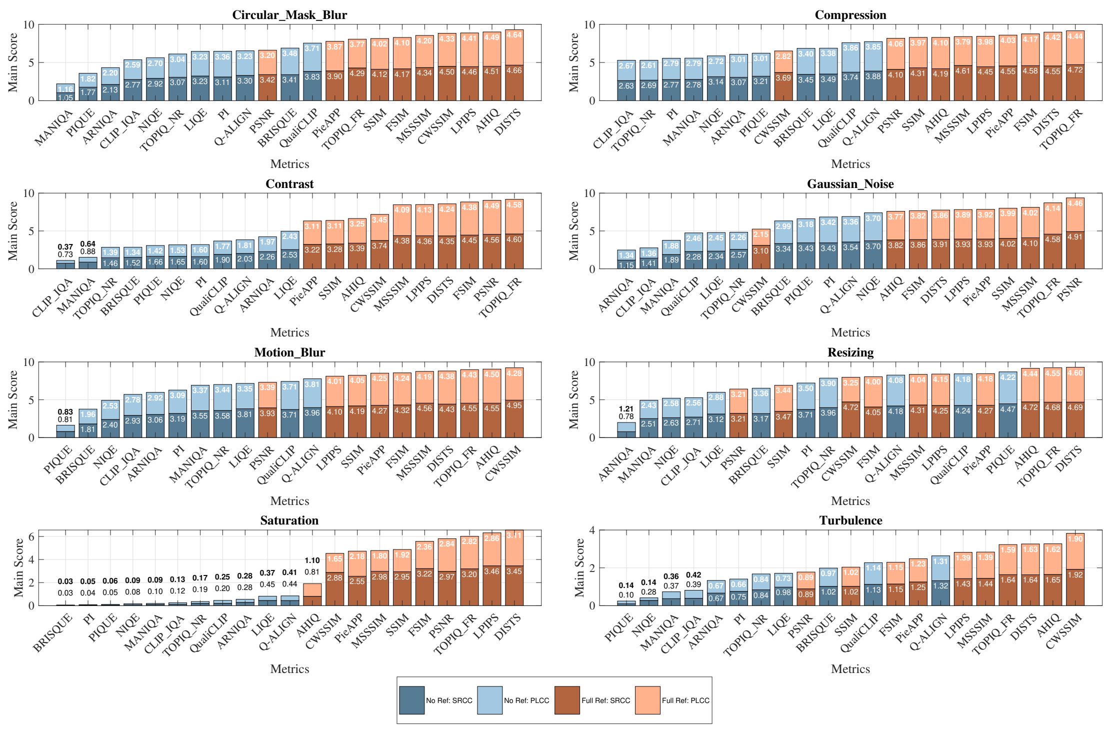


Figure 6.10: Correlation ($Main\ score = PLCC + SRCC$) between each metric and each distortion used (Zoom in for best viewing).

As with the dataset results, the FR metrics are consistently able to outperform the NR metrics, with the slight exception in the case of resizing distortion, where the NR metrics are able to perform on par with most FR metrics.

Table 6.8 shows that DISTS is the best performing metric (Overall and FR), with Q-ALIGN providing the best performance of the NR metrics. The lowest correlation (Overall and NR) is that of CLIPIQA, with SSIM providing the the poorest FR correlation.

Table 6.8: The individual ranking of each metric on each distortion (Based on main score). The sum (and re-ranking) of these therefore provide an idea of which metric is best suited across **all** distortions (The lower the better).

	Metric	Circular Mask Blur	Compression	Contrast	Gaussian Noise	Motion Blur	Resizing	Saturation	Turbulence	Overall Ranking
NR	ARNIQA	19 (9)	16 (6)	12 (2)	21 (11)	17 (7)	21 (11)	13 (3)	17 (7)	18 (7)
	BRISQUE	11 (2)	13 (4)	18 (8)	14 (5)	20 (10)	15 (6)	21 (11)	12 (3)	14 (5)
	CLIP_IQA	18 (8)	21 (11)	21 (11)	20 (10)	18 (8)	18 (8)	16 (6)	18 (8)	21 (11)
	LIQE	15 (5)	12 (3)	11 (1)	17 (7)	13 (3)	17 (7)	12 (2)	14 (4)	13 (3)
	MANIQA	21 (11)	18 (8)	20 (10)	19 (9)	15 (5)	20 (10)	17 (7)	19 (9)	20 (10)
	NIQE	17 (7)	17 (7)	16 (6)	10 (1)	19 (9)	19 (9)	18 (8)	20 (10)	19 (8)
	PI	14 (4)	19 (9)	15 (5)	12 (3)	16 (6)	13 (5)	20 (10)	16 (6)	15 (4)
	PIQUE	20 (10)	15 (5)	17 (7)	13 (4)	21 (11)	4 (1)	19 (9)	21 (11)	17 (9)
	Q-ALIGN	13 (3)	10 (1)	13 (3)	11 (2)	10 (1)	9 (3)	11 (1)	7 (1)	11 (1)
	QualiCLIP	10 (1)	11 (2)	14 (4)	18 (8)	11 (2)	6 (2)	14 (4)	10 (2)	12 (2)
	TOPIQ_NR	16 (6)	20 (10)	19 (9)	16 (6)	14 (4)	12 (4)	15 (5)	15 (5)	16 (6)
FR	AHIQ	2 (2)	7 (7)	8 (8)	9 (9)	2 (2)	3 (3)	10 (10)	2 (2)	3 (4)
	CW-SSIM	4 (4)	14 (10)	7 (7)	15 (10)	1 (1)	11 (8)	9 (9)	1 (1)	8 (7)
	DISTS	1 (1)	2 (2)	4 (4)	7 (7)	4 (4)	1 (1)	1 (1)	3 (3)	1 (1)
	FSIM	6 (6)	3 (3)	3 (3)	8 (8)	6 (6)	10 (7)	5 (5)	9 (8)	6 (6)
	LPIPS	3 (3)	5 (5)	5 (5)	6 (6)	9 (9)	7 (5)	2 (2)	6 (6)	4 (3)
	MSSSIM	5 (5)	6 (6)	6 (6)	3 (3)	5 (5)	8 (6)	7 (7)	5 (5)	5 (5)
	PieAPP	9 (9)	4 (4)	10 (10)	5 (5)	7 (7)	5 (4)	8 (8)	8 (7)	7 (8)
	PSNR	12 (10)	9 (9)	2 (2)	1 (1)	12 (10)	16 (10)	4 (4)	13 (10)	10 (9)
	SSIM	7 (7)	8 (8)	9 (9)	4 (4)	8 (8)	14 (9)	6 (6)	11 (9)	9 (10)
	TOPIQ_FR	8 (8)	1 (1)	1 (1)	2 (2)	3 (3)	2 (2)	3 (3)	4 (4)	2 (2)

The most striking finding from Figure 6.10 concerns saturation. It can be seen that the correlation of the FR metrics are significantly higher when compared to NR metrics. This is most likely due to the fact that a change in saturation is simply an image wide multiplication of pixel values, and no detail is being lost. Therefore the NR metrics are still able to identify the quality of the image, regardless of the level of saturation change. Whereas the FR metrics see this colour shift to be a large departure from the ground truth, and are able to correlate well.

Similar, but not quite as extreme results can be seen in the case of contrast. Unlike saturation, the changing of contrast removes detail in the form of 'clipping', where areas may be flattened out with access to only a narrow range of pixel values. However, as this distortion is mainly pixel value changes, the FR metrics are able to respond with much more accurate correlation. The difference in these two cases can be seen as either an advantage or disadvantage, depending on the application. In the case of saturation for example, if the image was being classified, the saturation level may not make a difference as the detail is still present in the image. On the other hand, if the goal was to produce a visually pleasing image, the NR metrics would be less able to identify the level of saturation. The best use of these results would be by combining the NR and FR metric responses. Again, for saturation, if an image is reported as high quality by a NR metric but low quality by FR, one could simply adjust the saturation as a post-processing step until a balance of the two metrics is found.

Another interesting outcome of these results concerns the distortion induced by image resizing. In this case, the NR metrics are interleaved with the FR metrics, providing competitive correlations. The likely reason for this is the fact that although the images become blurred, this blurring is essentially the same original image, but with a much lower sampling rate. This means that, on average, the difference between the distorted version and the ground truth is minimal, as the pixel values remain within a similar range. This therefore aligns well with the NR metrics understanding of 'good' vs 'bad' and the two types of metric provide similar correlations. In the case of compression induced distortion, which is a similar but distinctly different process to resizing, the difference between the two types of metrics can once again clearly be seen. This is because, unlike resizing, the compression process can actively change the range of pixel values far from the ground truth. This difference is therefore clearly caught by the FR metrics.

The distortions that actively alter the images are masked blur, motion blur, and Gaussian noise. These distortions are more 'natural', in that they are most likely to occur 'in camera' due to hand shake or camera setup. Overall, as before, the FR metrics are better correlated to the levels of distortion present. Promisingly, the NR metrics are still able to achieve high correlation scores in the presence of such distortions, as these cases are also most likely to be without ground truth data available.

Finally, the correlation of each metric with the turbulence distortion is the most insightful for this work. As expected the FR metrics provide good correlation, with CW-SSIM far outperforming the next 2nd best metric. This was as expected, as shown in [88], the CW-SSIM metric shows very good correlation with that of turbulent distortion. As with the blur and noise distortions, the identification of image quality through turbulence is most likely to occur without a ground truth present. So it is again promising that the Q-ALIGN NR metric is able to achieve a high correlation in this case.

However, as detailed previously, the fact that CW-SSIM and Q-ALIGN correlate positively with the Cn2 value of turbulence does not necessarily mean that the same will occur for the task of image quality analysis *after* turbulence mitigation, as the model may have imparted its own type of distortion that requires a different metric for best results.

On the whole, these results also show the advantage of data driven techniques, as found in DISTS, TOPIQ_FR and Q-ALIGN. However, unlike the dataset investigation, these results show more promising results for some more traditional metrics. CW-SSIM for example is found to provide the best correlation for both motion blur and turbulence. These distortions are similar, in that they both impart a form of motion onto the image. It is clear then that CW-SSIM is able to pick up on such distortion. The reason for this is due to the fact that CW-SSIM operates within the wavelet domain. This allows the metric to capture any local phase changes induced by the local motion of image content.

There are several final conclusions that can be drawn from the two analyses presented in this section. The first of which is that different metrics perform best on different image contents and distortion types, with no single metric emerging as universally optimal across all scenarios. This is extremely beneficial to identify, as it shows that there are situations in which some metrics may be more reliable than others. CW-SSIM for example provided high correlation with turbulence distortion, but was the worst performing FR metric for compressions and Gaussian noise distortions. It is therefore vital that any metric used in computer vision applications is appropriately chosen for the task at hand. Secondly, these results showed the clear distinction in performance between Full reference and no reference metrics. It was shown that, due to the clear differences in the levels of distortion present, the FR metrics were clearly able to correlate highly, above that of the NR metrics. However, it was shown that for more natural image contents and distortions the NR metric were able to produce competitive results. This is important, as these types of images are more likely to be obtained without the additional high resolution ground truth image, so is is therefore encouraging to see the ability of these metrics. Finally, it was shown in both scenarios, that each of the types of metric, NR and FR, have their limitiations. For example, it was shown that NR metrics struggle with the text based data of OTIS, whilst the FR metrics were less dominant on the resized images. This further shows that, when applying image quality metrics, the best idea of true image quality can only be achieved by making use of both types of metric.

6.5.3 Turbulence Mitigation: Observed Distortions

The purpose of Section 6.5.2 was to gain an understanding of the response of each metric to different types of distortions and image contents. Table 6.9 shows the average metric results for the ATSyn-static test dataset, where the results for both models are shown.

It was observed that the output images from models 'Sharp_1' and 'Sharp_2' tended to be of higher quality than that of 'Blurred_1' and 'Blurred_2', and it was reasoned that this was due to the perceptual loss present during training. This theory was further bolstered by the survey reported found in Section 6.5.1.

However, when analysing the results in Table 6.9 it would seem that the best performing model according to the majority of the metrics was 'Blurred_1', which corresponds to a DATUM model trained with a cosine scheduler. It can also be seen that in most cases where 'Blurred_1' performs the best, the second best model is 'Sharp_2', corresponding to a DATUM model trained with a cosine scheduler and a medium influence of perceptual loss. These results are very interesting, as from observations of the outputs of these two models, their images would be regarded as the most different to each other. What these results fully show is that there is a clear disconnect between different metrics, and no single metric should be wholly relied upon to provide a final judgement of image quality.

Table 6.9: Metric responses to each observed distortion in the ATSyn-static dataset model outputs. (Best results in **bold**, second best underlined). Also provided are the responses to the anomaly model outputs.

	Metric	Blurred_1	Blurred_2	Sharp_1	Sharp_2	Anomaly
	ARNIQA ↑	**0.556**	0.520	0.467	0.527	0.544
	BRISQUE ↓	42.221	44.454	38.189	**36.842**	*33.963
	CLIP_IQA ↑	**0.348**	0.317	0.224	0.284	0.300
	LIQE ↑	**2.171**	1.813	1.592	2.025	2.118
	MANIQA ↑	**0.261**	0.234	0.212	0.238	0.244
NR	NIQE ↓	4.905	5.147	4.527	**4.436**	*3.790
	PI ↓	5.872	6.375	5.682	**5.220**	*4.506
	PIQUE ↓	75.030	78.264	**53.945**	55.821	*37.727
	Q-ALIGN ↑	**2.709**	2.448	2.313	2.620	*2.723
	QualiCLIP ↑	0.447	0.417	0.384	**0.474**	*0.529
	TOPIQ_NR ↑	**0.328**	0.292	0.274	0.317	*0.333
	AHIQ ↑	**0.374**	0.337	0.331	0.367	*0.375
	CW-SSIM ↑	**0.977**	0.973	0.969	0.973	0.971
	DISTS ↓	0.172	0.192	0.146	**0.124**	*0.110
	FSIM ↑	**0.892**	0.886	0.886	0.892	0.891
	LPIPS ↓	0.272	0.321	0.291	**0.250**	*0.224
FR	MSSSIM ↑	**0.907**	0.901	0.896	0.902	0.899
	PieAPP ↓	1.568	1.850	1.870	**1.523**	*1.381
	PSNR ↑	**25.400**	25.130	24.890	25.133	24.965
	SSIM ↑	**0.756**	0.740	0.730	0.744	0.740
	TOPIQ_FR ↑	0.600	0.541	0.543	**0.610**	*0.638

* identifies the anomaly scores that would have outperformed the scores from the other models

Alongside the results of the four core models is that of an 'Anomaly' model. Results would suggest that the outputs of this model were not of good quality, and distorted with a repeating pattern of noise. However, of the NR metrics only ARNIQA, CLIPIQA, LIQE and MANIQA would have correctly predicted a lack in quality. All remaining NR metrics would have identified this unusual noise as high frequency content, and therefore scored it high. Of the FR metrics, the ones that would be led astray by this anomaly seem to also be the metrics that favoured the outputs of 'Sharp_2', therefore showing that these FR metrics also seem to promote high frequency content, regardless of its improvement (or lack of) of the image.

The other dataset under investigation was the Turbulence Text dataset, for which, two different tests were performed: one where 20 frames were provided to the model, and another where 60 frames were provided. For each of these, three models from Chapter 5 were found that provided three different levels of observed noise: low, medium and high. Due to the lack of ground truth for this dataset, only the NR metrics are examined.

It is worth highlighting that the following analysis of these images, and the corresponding metric responses, are solely regarding this additive noise, and not judging the 'perceptual quality' of the images. Without performing a suitable study/survey, the true perceptual quality of each image cannot be confidently ascertained, as it could be deemed that the images with the highest noise provides the best looking images, with the most readable text.

As can be seen from Table 6.10, most of the metrics agree for both the 20 frame outputs and the 60 frame outputs, with the exception of ARNIQUA, LIQE and QualiCLIP. This shows that the metrics are somewhat consistent between datasets, and will report the same quality no matter the noise source.

Table 6.10: No Reference metric responses to each level of observed noise in the Turbulence Text dataset model outputs for both 20 and 60 input frames, represented as "20/60". (**Bold** values indicate the best results per row). Also included are responses for the anomaly model outputs.

NR Metric	Low_Noise	Med_Noise	High_Noise	Anomaly (60)
ARNIQA ↑	**0.394**/0.405	0.386/0.320	0.378/**0.431**	0.386
BRISQUE ↓	44.235/45.686	43.354/50.852	**38.717**/**39.698**	43.111
CLIP_IQA ↑	**0.267**/**0.271**	0.185/0.190	0.192/0.190	0.221
LIQE ↑	1.035/**1.099**	1.028/1.032	**1.046**/1.029	1.062
MANIQA ↑	0.273/0.309	0.257/0.303	**0.317**/**0.337**	*0.371
NIQE ↓	6.488/6.224	**5.857**/**5.850**	9.277/8.272	12.021
PI ↓	7.909/7.323	7.568/6.608	**7.086**/**6.129**	8.419
PIQUE ↓	84.589/74.928	39.652/52.670	**20.916**/**29.597**	40.018
Q-ALIGN ↑	**1.983**/**2.121**	1.908/1.826	1.699/1.603	1.971
QualiCLIP ↑	0.229/**0.293**	**0.260**/0.245	0.214/0.214	*0.302
TOPIQ_NR ↑	0.234/0.274	0.219/0.235	**0.246**/**0.293**	*0.421

* identifies the anomaly scores that would have outperformed the scores from the other models

It is clear that the majority of the NR metrics reward high noise in the output images. Only Q-ALIGN is able to successfully decrease its score with each level of observed noise.

As with the ATSyn-static datset, the anomaly model was also tested on the Turbulence Text Dataset, with 60 frames as input. However, unlike the case for ATSyn-static, it does not wholly outperform the other outputs in terms of image quality metrics. This is most likely due to the nature of its distortion of uniform patterned noise which, when applied to the high contrast content of the text dataset results in images that are easier to identify as low quality by the NR metrics.

The results shown for the ATSyn-static Dataset and the Turbulence Text dataset are simply a comparison between a single observation of distortion and the metrics being investigated. They have shown various patterns in the behaviour of the metrics, but this study cannot provide a valid opinion without a more detailed study of perceptual quality for the images involved.

6.5.4 Turbulence Mitigation: Metrics for Application

The final study was that of comparing the metrics to real world applications such as classification. Using two different models and two different datasets, a full analysis of each metrics similarity to the classification could be undertaken. The resulting classifications of the two models for both datasets are shown in Table 6.11.

Table 6.11: Resulting classification accuracies of the DATUM and ED̄VR models on both the ‘turbulent imagenet’ subset and the Turbulence Text Dataset.

‘turbulent imagenet’

Model	**Top1**	**Top5**
DATUM	86%	97.44%
ED̄VR	93.48%	99.72%

Turbulence Text Dataset

Model	**CRNN**	**DAN**	**ASTER**
DATUM	90.4%	97%	97.15%
ED̄VR	10.73%	32.87%	40.13%

As can be seen, the ED̄VR model provides very poor classification accuracy when tested on the text data, but positive results on the ‘turbulent imagenet’ subset, whilst the DATUM model performs well on both. The models were chosen such that they represented a wide range of classification accuracies, therefore allowing the metrics to be tested for consistency over differing image qualities.

To identify the metrics that best correlate with classification accuracy, just as in Section 6.5.2, the PLCC and SRCC scores were used. However, instead of metric scores and distortion levels, these measured the correlation between the the metric scores and the classification accuracies. Therefore, if a metric scored poorly on incorrectly classified images, and good on correct classification, the correlation would be high.

Object Classification

The resulting correlations for each metric (from both models) on the 'turbulent imagenet' subset are shown in Figure 6.11.

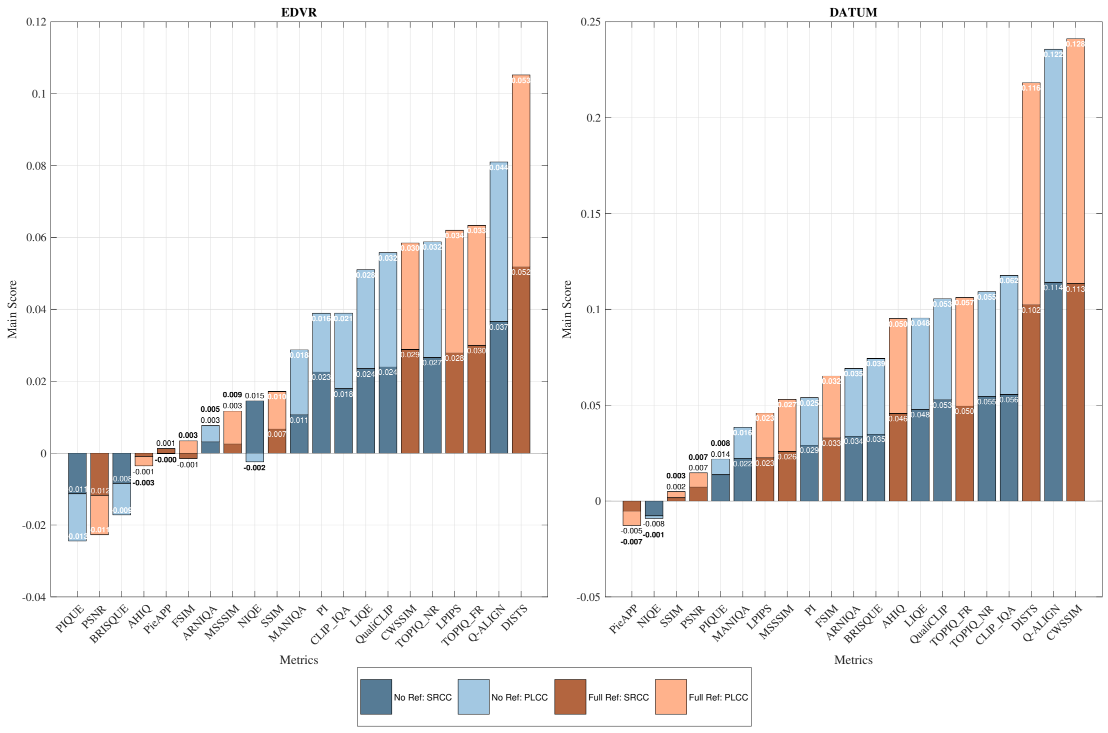


Figure 6.11: Correlation ($Main\ score = PLCC + SRCC$) between each metric and classification accuracy on the 'turbulent imagenet' subset of both the $\overline{\text{ED}}$VR model (trained on 'turbulence videos') and the DATUM model (trained on ATSyn-static) (Zoom in for best viewing).

The first insight to be gained from these results is the low correlation scores compared to those found in Section 6.5.2. This is due to the fact that the difference between incorrect and correct classification constitutes a far smaller margin. None of the metrics were able to fully distinguish between classifiable and unclassifiable images, therefore the best metric is the one that, on average, could separate the two the best. Another interesting outcome is that of negative correlations, such as PIQUE and PSNR for the E$\overline{\text{D}}$VR outputs, suggesting that the images that were incorrectly classified were of better quality. This agrees with what was initially proposed in this chapter, as it was shown that the PSNR metric would mislead in terms of useful image quality.

Unlike the results found in Section 6.5.2, for both models, the results for both FR and NR metrics are interleaved and there are good and bad correlations for both metric types.

For the E$\overline{\text{D}}$VR model outputs (which achieved an accuracy of 93.48%) the highest correlating metrics were that of DISTS and Q-ALIGN, Both of which significantly outperforming the next best of TOPIQ_FR. This was also the case for the outputs of the DATUM model (which achieved an accuracy of 86%), with the addition of CW-SSIM.

Interestingly, the correlation of the best metrics for the DATUM outputs were 2.3x greater than those from the E$\overline{\text{D}}$VR outputs. Alongside the 86% accuracy, this would suggest that the DATUM model produces images that occupy a wider range of image quality, where the images that are correctly classified are correspondingly of good quality, and the images that are unclassifiable are of worse quality, therefore allowing the metrics to better distinguish the difference. The E$\overline{\text{D}}$VR model however must produce images that are within a tighter distribution of quality, therefore the metrics struggle to fully distinguish between 'good' and 'bad' images. This can further be highlighted by examining the exact scores of a few selected metrics. Figures 6.12 and 6.13 show such values, where the scores have been ordered so that all the unclassifiable samples are plotted first, followed by the samples that were correctly classified (similar graphs for every metric can be found in Appendix C). Overlaid on both sets of samples is the average value for the incorrect/correctly classified samples respectively.

Firstly, these graphs further show that even the metrics with the best correlation are still not able to fully differentiate between correct and incorrect classifications. It is therefore the average response that must be examined. If on average a metric correlates with classification accuracy, then it is more likely to provide valuable information.

As well as the average score, the standard deviation is also a vital statistic for this analysis. If, for example, the standard deviation for correct samples is much smaller than that of incorrect samples, this means that the metric is more confident in its assessment when being shown correctly classified images.

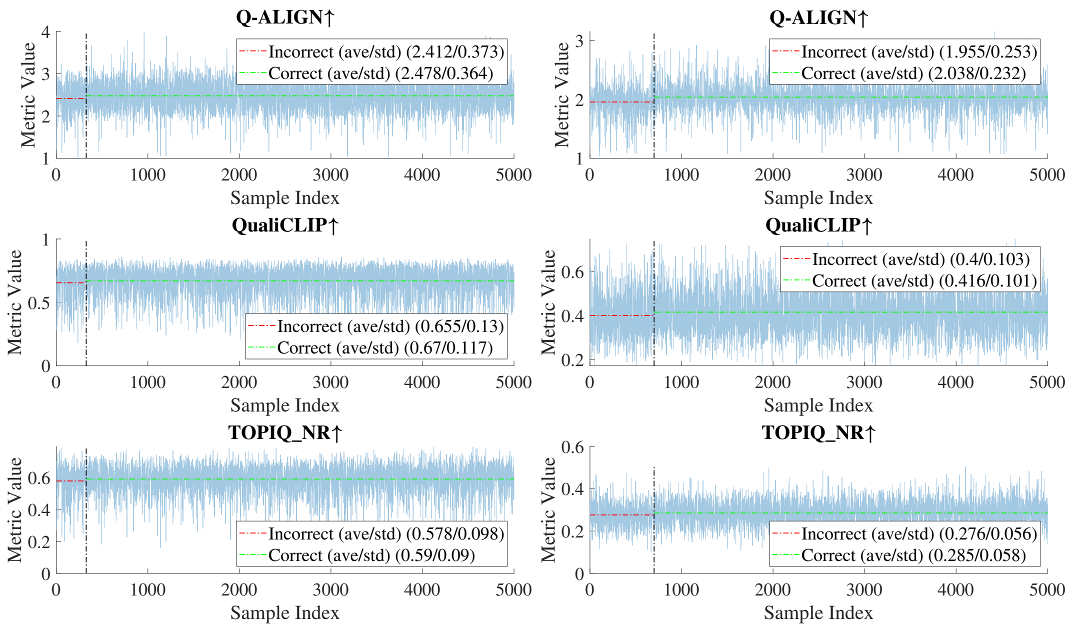


Figure 6.12: Scores of best 3 NR metrics on the turbulent_imagnet subset. Shown are the results given the E$\overline{D}$VR generated data (Left)(93.4%) and the DATUM generated data (right)(86%). Metric values are normalised between 0 and 1. Overlaid is the average metric score for samples with incorrect and correct classifications (Zoom in for best viewing).

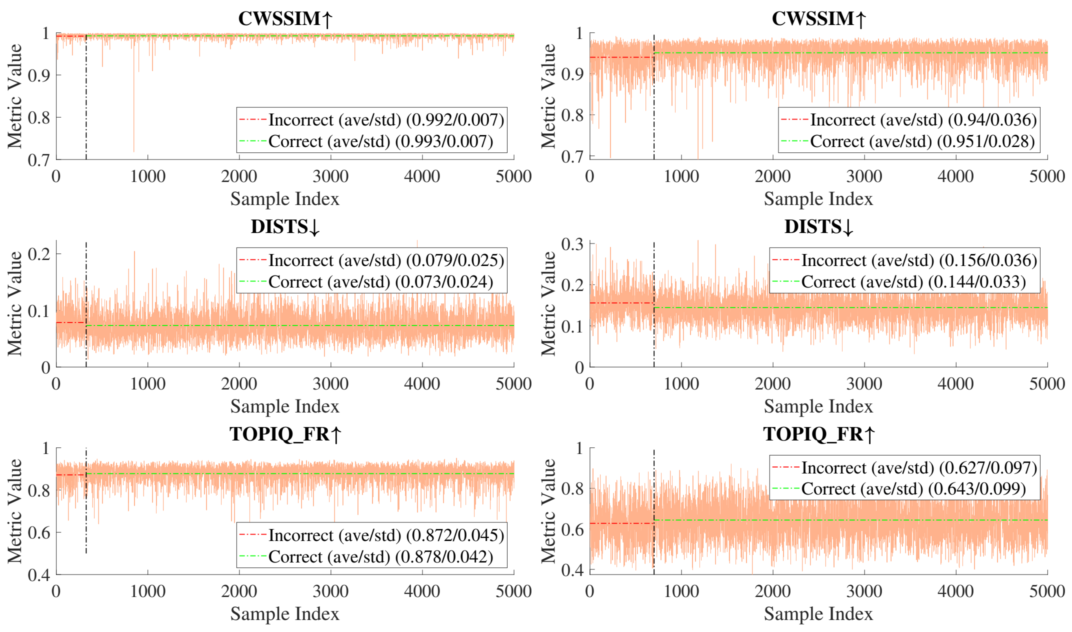
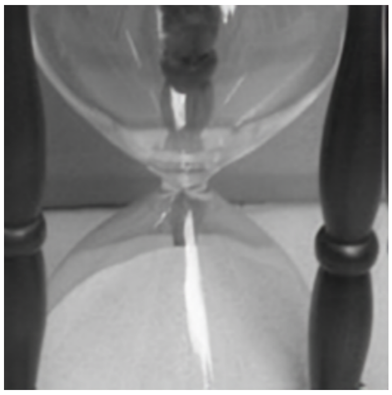


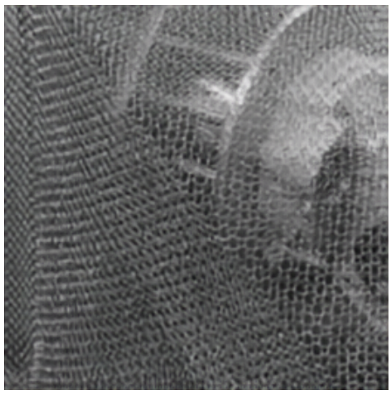
Figure 6.13: Scores of best 3 FR metrics on the turbulent_imagnet subset. Shown are the results given the E$\overline{D}$VR generated data (Left)(93.4%) and the DATUM generated data (right)(86%). Metric values are normalised between 0 and 1. Overlaid is the average metric score for samples with incorrect and correct classifications (Zoom in for best viewing).

(a) Correctly classified output image from E$\overline{\text{D}}$VR with a DISTS score of 0.0135

(b) Incorrectly classified output image from E$\overline{\text{D}}$VR with a DISTS score of 0.159



(c) Correctly classified output image from DATUM with a DISTS score of 0.0317

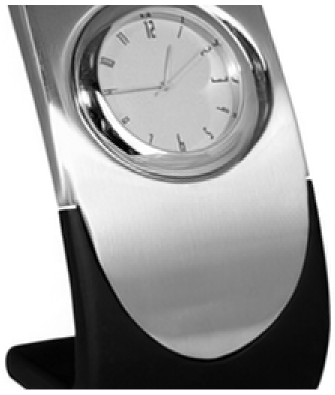


(d) Incorrectly classified output image from DATUM with a DISTS score of 0.299

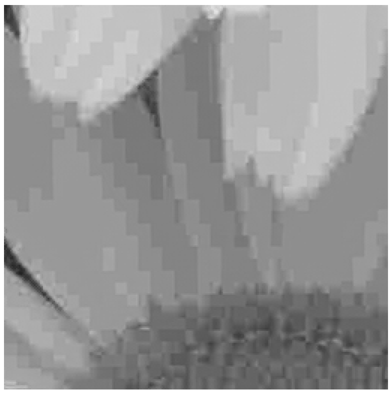
Figure 6.14: Comparison between images and their corresponding DISTS score. It is shown that for both E$\overline{\text{D}}$VR (Top) and DATUM (Bottom), the score correlated with the ability to correctly classify the image.

These graphs do clearly show how much the model outputs impact the metric responses. For example, in the case of E$\overline{\text{D}}$VR with a high classification accuracy, the CW-SSIM metric struggles to identify the difference between correct and incorrect classifications. However when applied to the DATUM model outputs, the standard deviation of the metric is far larger. This shows that the DATUM model is more varied in its output image quality, and the metrics respond as such.

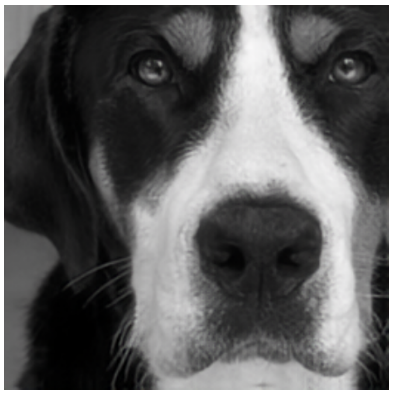
Given these results, it can be surmised that, of the NR metrics, Q-ALIGN best correlates with the 'turbulent imagenet' classification accuracy, and DISTS performs best for the FR metrics, where the high score of CW-SSIM is less trustworthy due to its large difference between model outputs. These results also align with those found in Section 6.5.1, where both metrics agreed with the outcome of the image quality survey. They were also found to be the most well



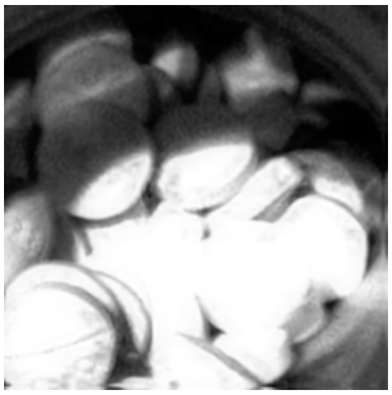
(a) Correctly classified output image from E$\overline{\text{D}}$VR with a Q-ALIGN score of 3.977



(b) Incorrectly classified output image from E$\overline{\text{D}}$VR with a Q-ALIGN score of 1.023



(c) Correctly classified output image from DATUM with a Q-ALIGN score of 3.152



(d) Incorrectly classified output image from DATUM with a Q-ALIGN score of 1.033

Figure 6.15: Comparison between images and their corresponding Q-ALIGN score. It is shown that for both E$\overline{\text{D}}$VR (Top) and DATUM (Bottom), the score correlated with the ability to correctly classify the image.

rounded metrics when tested on common image distortions in Section 6.5.2. In Section 6.5.3, DISTS was able to successfully identify what had been observed as 'sharp' and Q-ALIGN was able to accurately rank the levels of observed noise in the text data. Examples of the quality assessment of these two metrics can be seen in Figures 6.14 and 6.15. As with the previous results, the reasoning for these metrics success can be identified by analysing their methodologies, where in both cases their data driven nature allows them to infer a great deal with regards to image quality. Given this, the ability of CW-SSIM to identify turbulence, previously observed in Section 6.3, is also evident in this analysis.

A final logical step, given these results, is that of formulating a custom metric that aims to utilise the advantages of each of the best performing metrics. Given that DISTS, Q-ALIGN and CW-SSIM provided the best correlation scores, an investigation was undertaken to identify an optimal combination that would outperform each individually. This investigation led to the identification of a new metric, the Turbulence Quality Index (TQI) which is defined as,

$$TQI = \frac{(1 - DISTS) + CW - SSIM + \frac{Q\text{-}ALIGN}{5}}{3}, \tag{6.2}$$

where the opposite 'polarity' of DISTS is first solved by subtracting it from 1. The larger range of Q-ALIGN is solved by a division, and the average of these metrics summed makes up the final metric. Figure 6.16 shows that this new metric obtains a better total score than all original metrics for both the EDVR and DATUM outputs, where the correlation score for the DATUM data is significantly higher. This difference is simply due to the presense of CW-SSIM within the TQI metric. As shown in Figure 6.11, CW-SSIM does not perform as well as the others for the EDVR data, whilst all three metrics perform well for the DATUM data. Its presence therefore reduces the capability on the EDVR data, but is a worthwhile addition nonetheless. As well as the correlation scores, the individual values of TQI can also be plotted. Such is shown in Figure 6.17, where it can be seen that even with an improved correlation, this new metric is still unable to fully differentiate between correct and incorrect classifications. It does however provide an improvement over the metrics shown in Figures 6.12 and 6.13.

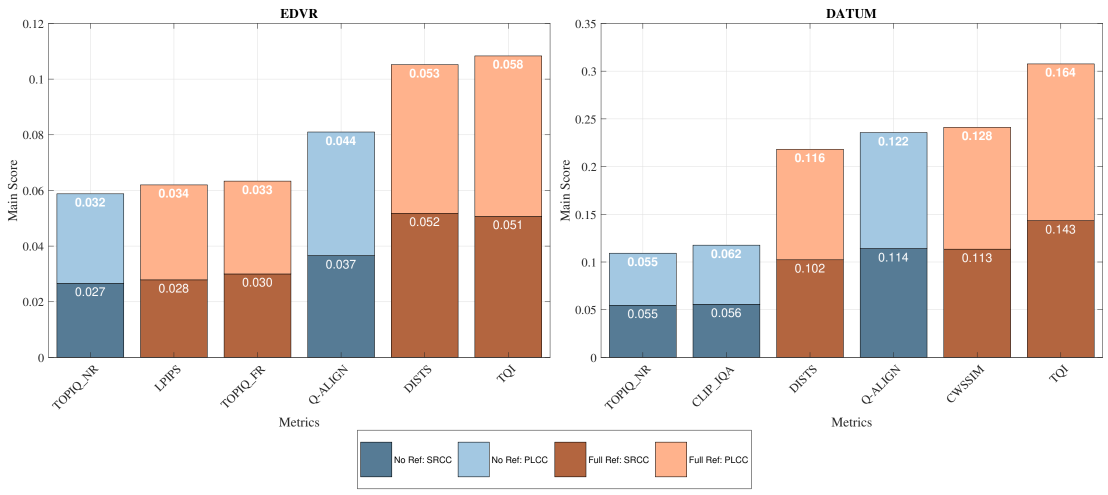


Figure 6.16: Correlation ($Mainscore = PLCC+SRCC$) between the top performing metrics and classification accuracy. Now including the newly defined TQI metric, that is shown to outperform original metrics. Results are based on the subset of 'turbulent imagenet' for both the E$\overline{\text{D}}$VR model (trained on 'turbulence videos') and the DATUM model (trained on ATSyn-static) (Zoom in for best viewing).

Due to the presence of both DISTS and CW-SSIM, the TQI metric is inherently a FR metric. When investigating into the use of metric combinations for

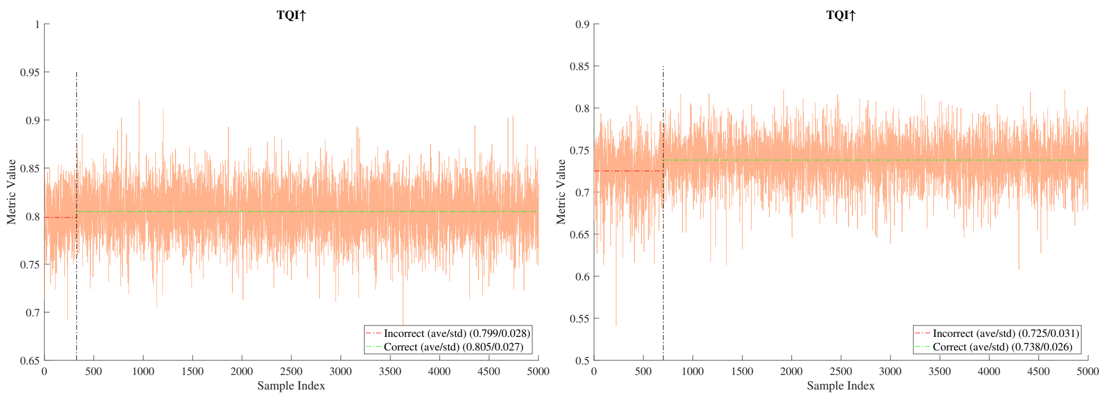


Figure 6.17: Scores from *TQI* on the ‘turbulent imagenet’ subset. Shown are the results given the E$\overline{D}$VR generated data (Left)(93.4%) and the DATUM generated data (Right)(86%). Metric values are normalised between 0 and 1. Overlaid is the average metric score for samples with incorrect and correct classifications. (Zoom in for best viewing).

NR, no combination was able to outperform the raw results of Q-ALIGN. Therefore, it is suggested that if ground truth images are indeed available, the *TQI* metric will provide the most accurate results. If no ground truth is available, Q-ALIGN by itself is the best option.

Text Classification

The correlations of each metric on the Turbulence Text dataset is shown in Figure 6.18, where, for each model, three different text classification models were utilised. As with the results in Section 6.5.3, due to the lack of ground truth for this dataset, only NR metrics are examined.

For the classification results on the E$\overline{D}$VR model outputs, it can be seen that the trend in correlation is the same for each of the text classification models, even when the different classifiers produce varied accuracy results. This is also mostly true with the DATUM outputs, with some changes to the order for each text classification model. For the E$\overline{D}$VR outputs for example, there is a 30% classification accuracy increase between CRNN and ASTER, but correlation order of the metrics is the same. This therefore shows that, when a better classifier is used, the correlation behaviour of the metrics remains consistent, therefore implying that, any metrics that are found to have good correlation with text classification will perform well, regardless of the classifier model being used.

From the results shown in Figure 6.18, it can clearly be seen that the correlations of the metrics vary greatly between the two different models. Not only this, but, another difference between the two models can also be observed in the behaviour of the metrics where, for E$\overline{D}$VR, the metrics seem to linearly decrease

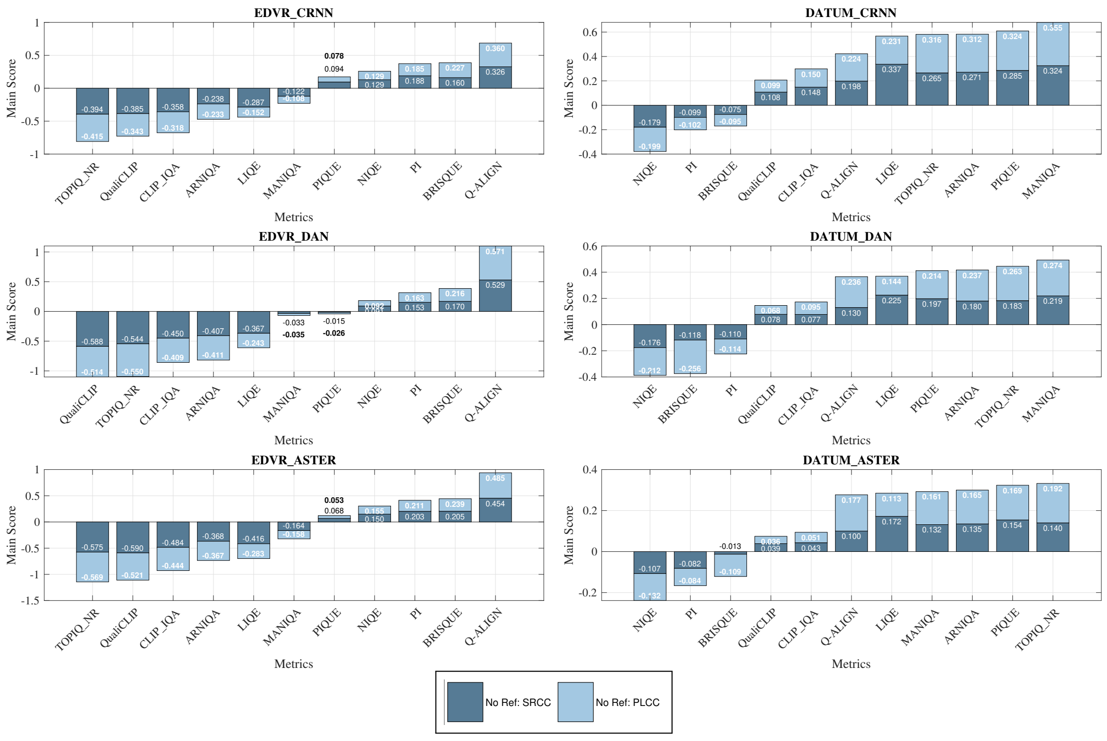


Figure 6.18: Correlation ($Main\ score = PLCC + SRCC$) between each metric and each text classification model accuracy on the Turbulence Text dataset. For both the EDVR model (trained on 'turbulence videos') and the DATUM model (trained on ATSyn-static) using the CRNN, DAN and ASTER text classification models (Zoom in for best viewing).

in performance from Q-ALIGN down to TOPIQ_NR, whereas for DATUM, six of the metrics are able to provide high correlation (MANIQA, PIQUE, TOPIQ_NR, ARNIQA, LIQE & Q-ALIGN) before a sudden drop off of in quality. The best way in which to analyse these results is by plotting the metric scores alongside the classification accuracies. The best metrics from each model is shown in Figures 6.19 and 6.20 (for similar graphs for all NR metrics, see Appendix C).

From these plots, a few key insights and observations can be made. Firstly, as with the results on the 'turbulent imagenet' subset, it can be seen that even the best correlating metrics are unable to fully distinguish between correct and incorrect classifications, with each metric assigning both high and low image quality values to both correctly and incorrectly classified images. These plots do show however that, on average, these metrics could be used to provide a prediction of text classification accuracy.

However, as a caveat to this point, due to the overall classification performance of each model, the number of samples in either the 'correctly classified' or 'incorrectly classified' bins will vary greatly. The incorrect classifications for the ASTER classifier on the DATUM outputs (Figure 6.20) for example are only

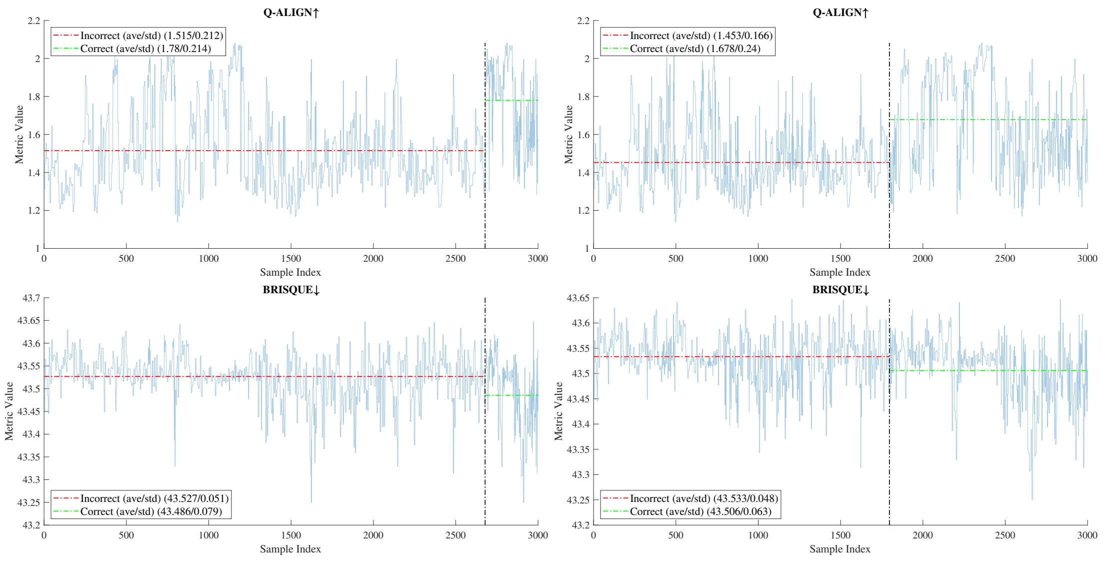


Figure 6.19: All values for the best correlating metrics for images from the E$\overline{\text{D}}$VR Model: Q-ALIGN (Top) and BRISQUE (Bottom). Ordered with resepct with classification ability from CRNN (Left) and ASTER (Right). (Zoom in for best viewing).

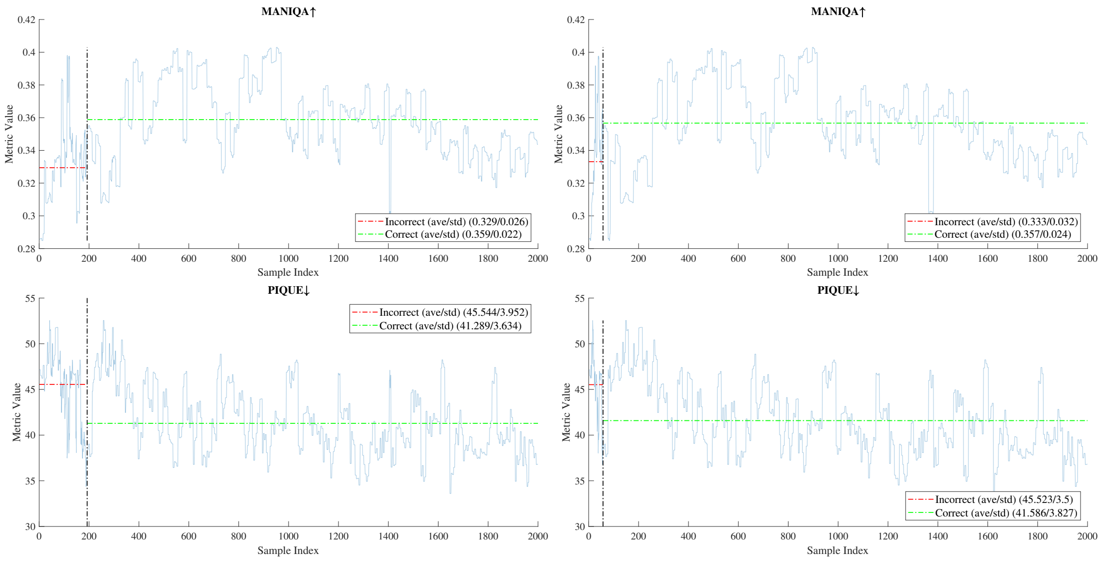


Figure 6.20: All values for the best correlating metrics for images from the DATUM Model: MANIQA (Top) and PIQUE (Bottom). Ordered with respect with classification ability from CRNN (Left) and ASTER (Right). (Zoom in for best viewing).

represented by 57 samples (vs 1943 correct samples). This makes the comparison between the average metric scores of correct and incorrect samples to be approached with caution.

A final important aspect of these results worth highlighting is that of the y-axis scale. For the BRISQUE results shown in Figure 6.19, the range of the y-axis only spans between 43.2 and 43.7. This is important, as the range of the BRISQUE metric is between 0 and 100. Therefore, the difference here between 'correct' and 'incorrect' is actually extremely small. However, the correlation scores do not take the potential range into account, and see the small change in BRISQUE between classifications and mark it highly in response.

For the E$\overline{\text{D}}$VR model outputs, it can be seen in Figure 6.19 that the Q-ALIGN metric mostly correlates well with classification accuracy, where, on average, it correctly assigns the correctly classified images with a higher image quality score whilst also, on average, assigning lower scores to incorrectly classified images. However it is not fully reliable, as it assigns low scores to correctly classified images and vice versa.

For the second best correlating metric, BRISQUE, a clear difference can also be seen. However, as previously mentioned, this difference is only a matter of $\sim$0.1 and, for a metric with a range of 100, this small difference would be less likely to be noticed, therefore reducing its potential effectiveness as a text classification image quality replacement.

The best scoring metrics for the DATUM outputs illustrated in Figure 6.20 both show positive correlations. The downside to these results however, as also mentioned previously, is the lack of samples of incorrectly classified images. Therefore, for the misclassified samples, the corresponding metric response could be attributed to random chance. This being said it can be observed, that the MANIQA metric is mostly consistent in its opinion of good vs bad, where it is quite clearly scoring the misclassified samples lower than that of the correctly sampled.

As noted earlier, the correlation of metrics is not consistent between the different model outputs. To further highlight this, it can be seen that certain metrics actually reverse their correlations between models. One such metric is BRISQUE, which ranks second-best on the E$\overline{\text{D}}$VR data, but falls to the bottom three for the DATUM data. Similarly, MANIQA performs best on the DATUM data, but shows a negative correlation with the E$\overline{\text{D}}$VR data. Again, this can clearly be seen by plotting the metrics and comparing with classification. Shown in Figure 6.21 is the apparent reverse of correlation of the BRISQUE and MANIQA metrics. In the case of BRISQUE, it showed good correlation for E$\overline{\text{D}}$VR, however reversed for the DATUM outputs. It can be seen that the reason for this reversal is most likely due to the instability caused by lack of sample size, where due to a few outliers, the average metric of the incorrect samples is brought below that of the correct samples.

In the case of MANIQA, the reason for the reversal from DATUM to E$\overline{\text{D}}$VR is less clear. Therefore, to help gain an even better understanding of this behaviour,

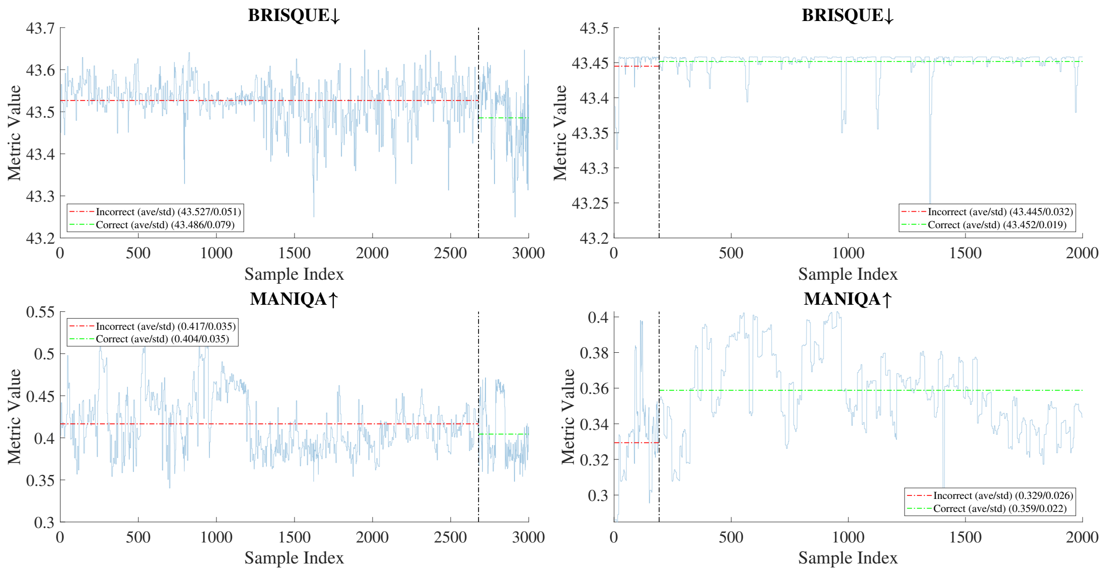
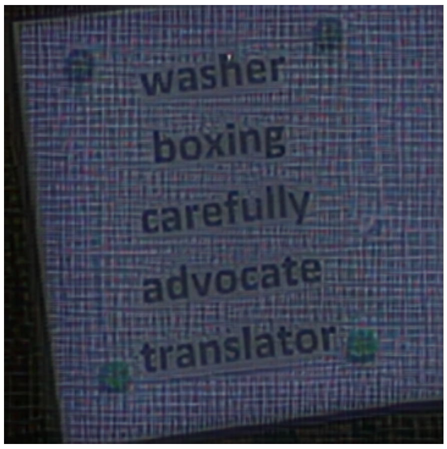


Figure 6.21: Values for the BRISQUE (Top) and MANIQA (Bottom) metrics, ordered with respect with classification ability from CRNN. Noting the correlation reversal between the E$\overline{\text{D}}$VR (Left)(10.73%) and DATUM (Right)(90.4%) output images. (Zoom in for best viewing).

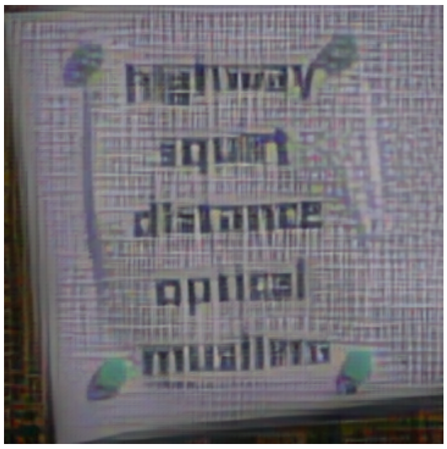
an example of each case is illustrated in Figure 6.22. It is clear to see that the outputs from the DATUM model are of far higher quality than that from the E$\overline{\text{D}}$VR model, even in the case of a low MANIQA score. Interestingly the highest score is given to the image in Figure 6.22d which, from visual analysis would be the hardest in which to identify text. This shows that the MANIQA metric would not be a good candidate for a text classification image quality metric, as it is less useful in highlighting the sharp edges of clear text, while it more suited to promoting high frequency noise. This was also observed in Section 6.5.3.

As well as metrics that change their correlation between models, there also exist metrics that have unusually high *negative* correlation, such as CLIPIQA, QualiCLIP and TOPIQ_NR for the E$\overline{\text{D}}$VR model outputs. The response of these metrics is illustrated in Figure 6.23, where it can clearly be seen that each metric assigns worse image quality to those images that have been classified correctly, and vice versa. This suggests that these metrics are looking for the inverse of what could be defined as ‘clear text’. As with the MANIQA results, this can be seen clearer with examples, as shown in Figure 6.24. These examples show that the metrics seem to promote images with a higher level of noise in them, and view the ‘clearer’, ‘less noisy’ images as lower quality. This is an interesting observation, as with this knowledge, these metrics could be utilised in reverse, to find images that would *not* be classified well due to a high level of observed noise.

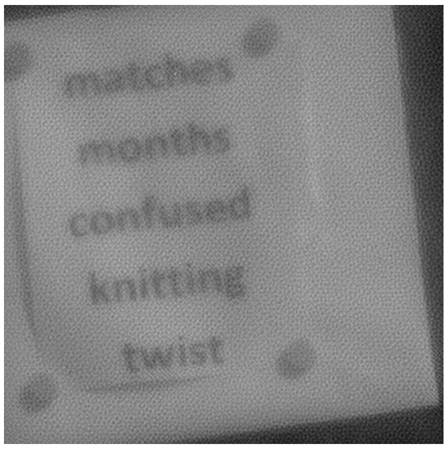
Of all the NR metrics tested, only Q-ALIGN was able to consistently correlate



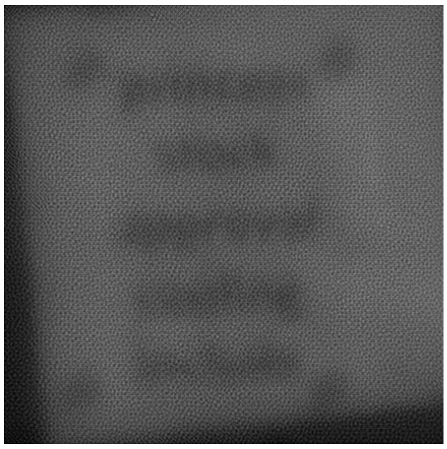
(a) Correctly classified (5/5 Correct) output image from DATUM with a MANIQA score of 0.403



(b) Incorrectly classified (0/5 Correct) output image from DATUM with a MANIQA score of 0.286



(c) Correctly classified (4/5 Correct) output image from E$\overline{\text{D}}$VR with a MANIQA score of 0.348



(d) Incorrectly classified (0/5 Correct) output image from E$\overline{\text{D}}$VR with a MANIQA score of 0.528

Figure 6.22: Comparison between the classification of text and the corresponding MANIQA score. It is shown that for DATUM (Top), the metric correlates with classification, and that for E$\overline{\text{D}}$VR (Bottom) a reverse correlation is found.

positively on both models outputs, with the responses shown in Figure 6.25, as well as image examples shown in Figure 6.26. These show that Q-ALIGN appears to promote images that contain clear text, and is not 'fooled' by images with high observed noise. Although it is not foolproof (as shown by the anomaly results in Section 6.5.3), it is Q-ALIGN that would best be able to provide a somewhat reliable measure of image quality for text classification. As with previous results, the likely reason for the quality of Q-ALIGN stems from its use of an LLM for producing image quality scores. This alongside a ViT architecture allows the metric to gain a solid understanding of the image contents, as well as a detailed and flexible 'score' in the form of the LLM output.

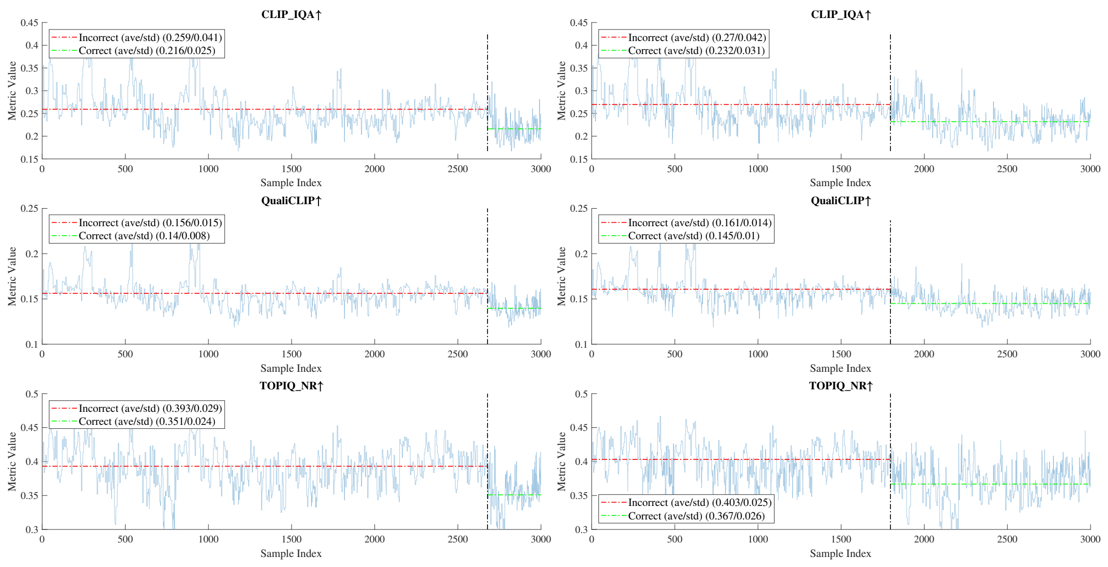
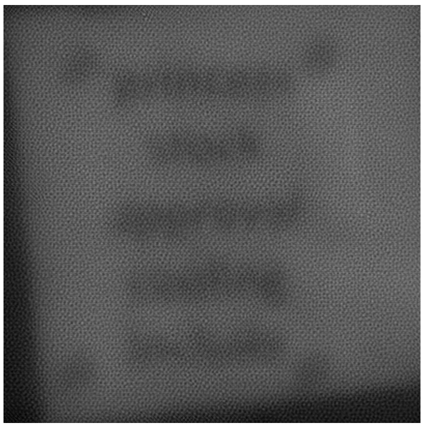


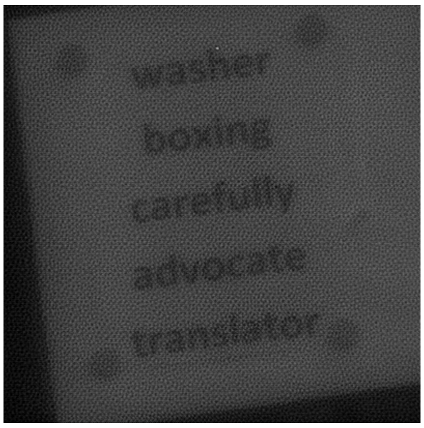
Figure 6.23: Values for the metrics that display negative correlation on the EDVR output images: CLIP_IQA (Top), QualiCLIP (Centre) and TOPIQ_NR (Bottom). Ordered with respect with classification ability from CRNN (Left) and ASTER (Right). (Zoom in for best viewing).

As with 'turbulent imagenet', it is logical to make use of these results and attempt to create a metric that is able to overcome any weaknesses presented. With regards to these results however, it is not as trivial as in the case for 'turbulent imagenet' due to the erratic nature of the metrics between models, where the simple combination of metrics will not provide any improvement. In fact, it is this erratic nature that can be utilised to create a metric that will perform well no matter the model being used. As mentioned previously, the reason for metrics 'reversing' their correlation scores is that of the image sources, where the DATUM model was able to produce better quality images. In this case, it was MANIQA that was best able to identify if text would be classified correctly or not. For the lower quality outputs of EDVR, it was Q-ALIGN that performed best, with MANIQA struggling to identify the good vs the bad. This simply shows that the MANIQA metric is simply better when the images are of better quality, and Q-ALIGN is better when the images are of poorer quality. By harnessing this information, a new metric can be created that 'selectively chooses' which metric to use, where the choice is driven by the results of other metrics.

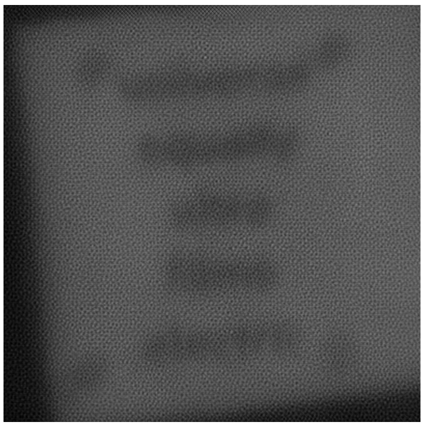
It was therefore identified that the comparison between MANIQA and Q-ALIGN (normalised) led to the best resulting metric. Shown in Figure 6.27 are the metric scores from both Q-ALIGN and MANIQA. It can clearly be seen that when evaluated on the EDVR outputs, the Q-ALIGN score is mostly below that of MANIQA (2,955/3,000). When presented with the DATUM outputs however,



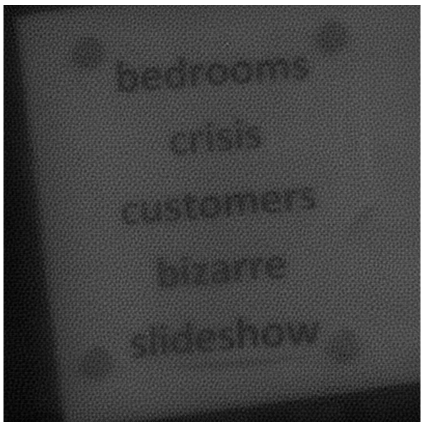
(a) Incorrectly classified (0/5 Correct) output image from EDVR with a CLIPIQA score of 0.428



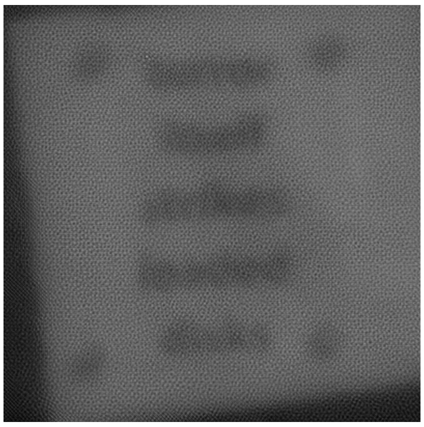
(b) Correctly classified (4/5 Correct) output image from EDVR with a CLIPIQA score of 0.167



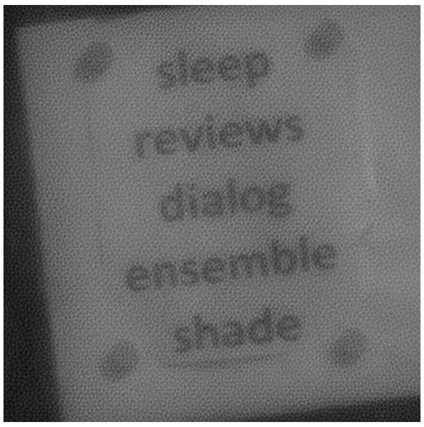
(c) Incorrectly classified (0/5 Correct) output image from EDVR with a QualiCLIP score of 0.22



(d) Correctly classified (3/5 Correct) output image from EDVR with a QualiCLIP score of 0.119



(e) Incorrectly classified (0/5 Correct) output image from EDVR with a TOPIQ_NR score of 0.462



(f) Correctly classified (3/5 Correct) output image from EDVR with a TOPIQ_NR score of 0.301

Figure 6.24: Examples of cases where metrics present negative correlations. Such metrics were CLIPIQA (Top) QualiCLIP (Centre) and TOPIQ_NR (Bottom).

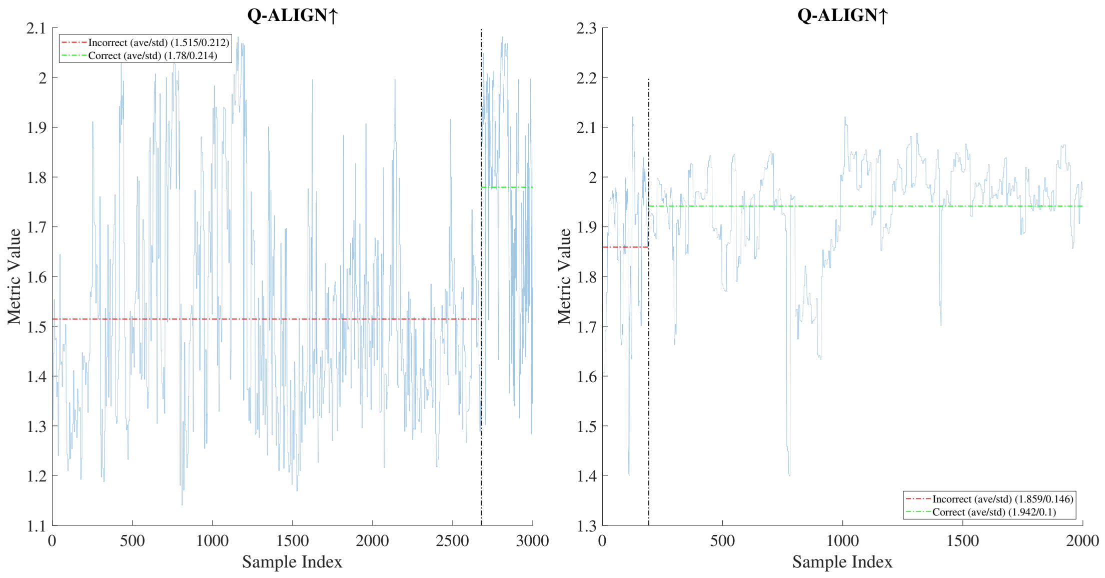
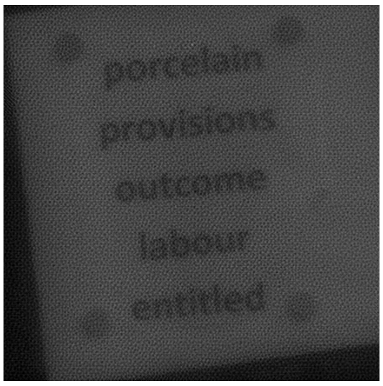


Figure 6.25: Values for the Q-ALIGN metric, which was one of the only metrics that was able to correlate positively over both models, ordered with respect with classification ability from CRNN. EDVR (Left)(10.73%) and DATUM (Right)(90.4%). (Zoom in for best viewing).

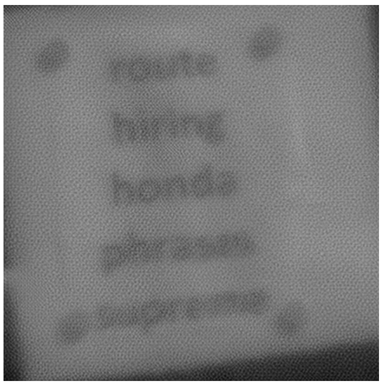
the Q-ALIGN score is mostly higher than MANIQA (1,630/2,000). Using this information, a new metric was developed, the Turbulence Quality Index for Text (TQI_T), and is simply defined as,

$$TQI_T = min\left(MANIQA, \frac{Q\text{-}ALIGN}{5}\right) \tag{6.3}$$

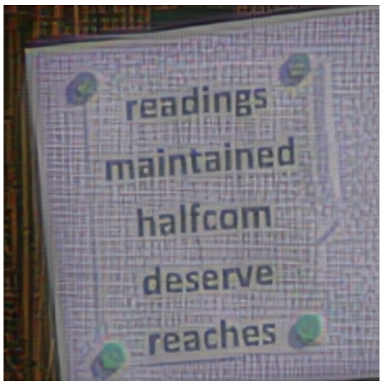
The resulting correlation scores and output values of this new metric are shown in Figures 6.28 and 6.29 respectively. These show that TQI_T outperforms all other metrics on the DATUM outputs, and matches that of Q-ALIGN for the EDVR outputs. This is as expected, as the metric is designed such that it is able to operate one both distributions of data. The key to this success is the response of the MANIQA metric. As shown in Figure 6.22, the EDVR model generates images with a higher base level of noise when compared to DATUM. This is reflected in the MANIQA score, which rewards such high frequency content. This therefore leads to the average MANIQA score being higher than that of Q-ALIGN. This indication of noise provided by MANIQA allows a user to make the choice of Q-ALIGN as a more appropriate metric. In the case of the DATUM data however, the base noise is reduced, and therefore results in lower MANIQA scores. In such a case, as shown in Figure 6.18, the lack of noise allows MANIQA to provide more reasonable scores, which result in a better correlation with classification accuracy. The MANIQA score therefore acts as a 'noise detector' that allows



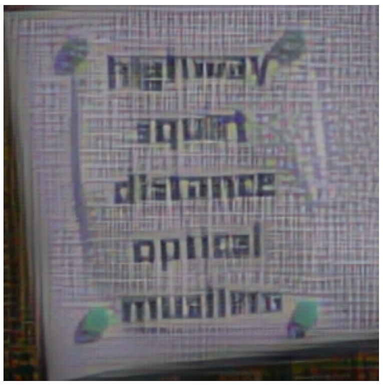
(a) Correctly classified (4/5 Correct) output image from E$\overline{\text{D}}$VR with a Q-ALIGN score of 2.082



(b) Incorrectly classified (0/5 Correct) output image from E$\overline{\text{D}}$VR with a Q-ALIGN score of 1.141



(c) Correctly classified (3/5 Correct) output image from DATUM with a Q-ALIGN score of 2.121



(d) Incorrectly classified (1/5 Correct) output image from DATUM with a Q-ALIGN score of 1.606

Figure 6.26: Example images where Q-ALIGN presents positive correlation for both E$\overline{\text{D}}$VR (Top) and DATUM (Bottom).

the TQI_T metric to identify the characteristics of the images, and apply the best metric accordingly.

Thresholds for Classification

The final step of this investigation was to determine if any of these metrics could be used 'in place' of classification, therefore providing a measure of classification probability. To identify this, the Balanced Accuracy (BAcc) of the predictions at different thresholds was evaluated. Using Eq. (3.17), the optimal threshold, τ^*, for each metric was found for each dataset/model scenario.

Values of the BAcc below 50% correspond to *negative* balance, therefore indicating that the distributions are inverted (i.e. the metric scores good images with a bad score, and vice versa). As seen from the correlation results, some

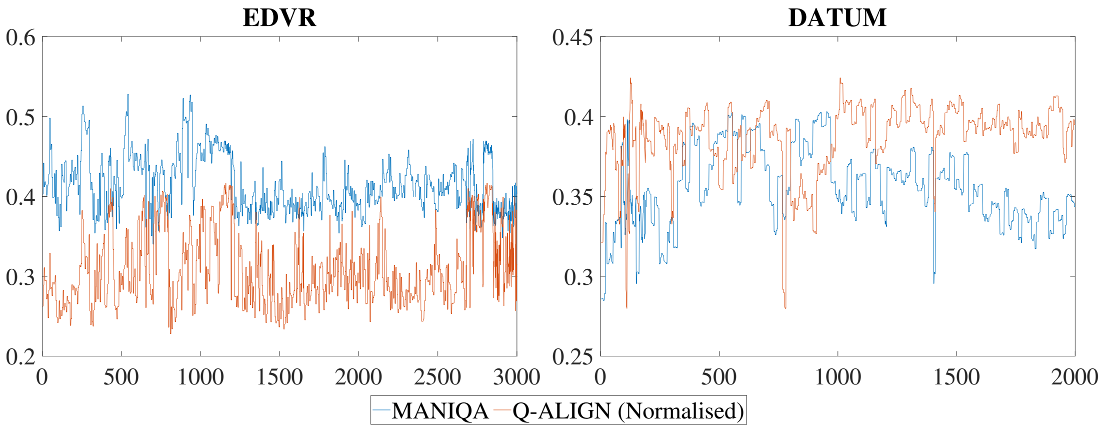


Figure 6.27: Comparison between the scores from MANIQA and Q-ALIGN. Showing that, for the EDVR outputs, the Q-ALIGN score is mostly below that of MANIQA (98.5% of the time). For the DATUM outputs however, the Q-ALIGN score is higher on average (81.5% of the time).

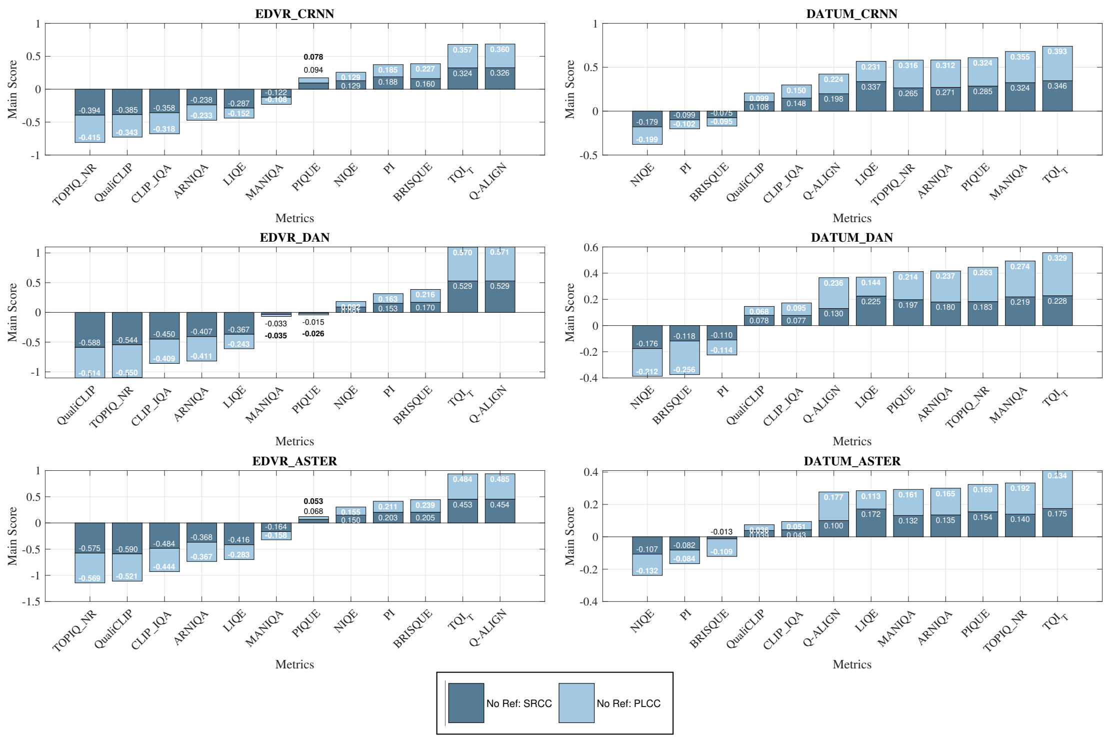


Figure 6.28: Correlation ($Main\ score = PLCC + SRCC$) between each metric and each text classification model accuracy on the Turbulence Text dataset, now including TQI_T. For both the $\overline{\text{EDVR}}$ model (trained on 'turbulence videos') and the DATUM model (trained on ATSyn-static) using the CRNN, DAN and ASTER text classification models (Zoom in for best viewing).

metrics do indeed show negative correlation scores. Therefore, when evaluating the BAcc, if a metric indicated negative correlation, the polarity of the metric was flipped, such that all BAcc values are above 50%. This then allows a comprehensive comparison of the metrics, as some of the negative correlated results

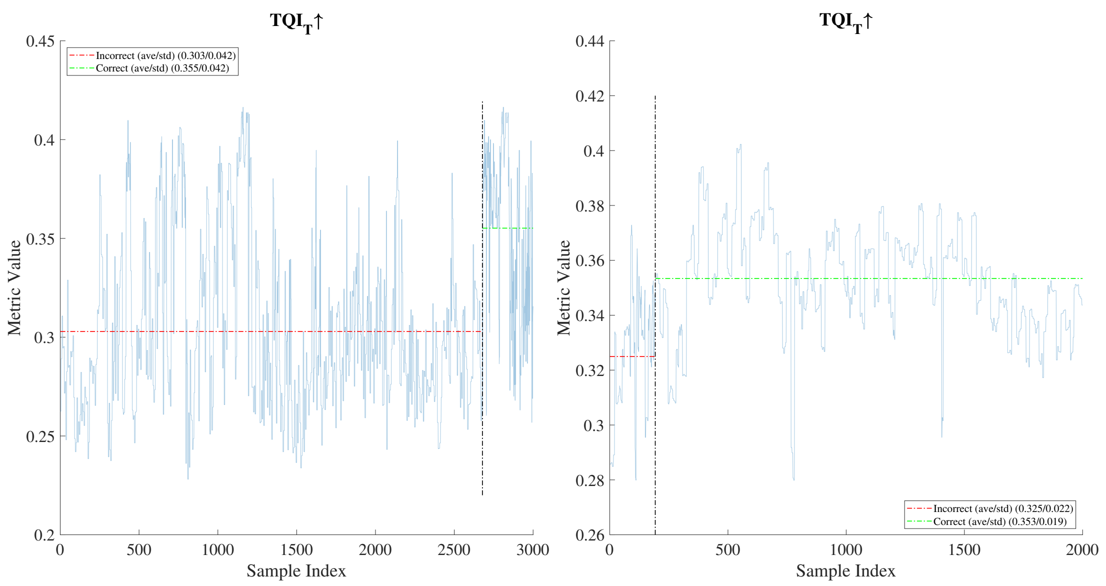


Figure 6.29: Values for the proposed metric, TQI_T, ordered with respect with classification ability from CRNN. E$\bar{\text{D}}$VR (Left)(10.73%) and DATUM (Right)(90.4%). (Zoom in for best viewing).

are still strongly correlated, allowing the potential of a metric used in reverse to provide an image classification prediction.

The resulting BAcc scores for the 'turbulent imagenet' subset are shown in Tables 6.12, where the metrics that presented negative correlation are marked with a *.

The results in Table 6.12 align with those found via correlation, with Q-ALIGN providing the best predictions of classification accuracy for NR metrics, alongside TOPIQ_NR. As for the FR metrics, it is TQI, DISTS and CW-SSIM that provide the best predictions. This is again as expected, and in line with the previous correlation scores. There are two main takeaways from these results. First to notice is the disparity between the results of the two models. For each metric, the threshold and the corresponding Balanced Accuracy is never consistent over both models. This indicates that, whilst a threshold might provide good results for one model, that same threshold would result in a significant decrease in accuracy for another. For a majority of the metrics, the BAcc scores for the DATUM model are higher than E$\bar{\text{D}}$VR. This again indicates that the DATUM model is the poorer of the two, as E$\bar{\text{D}}$VR is able to consistently produce higher quality images, therefore making it harder for the metrics to distinguish between classifiable and unclassifiable images. An initial solution to the difference in thresholds could be to simply use the median of the two reported thresholds. However, this would result in a decrease in accuracy for both models. For exam-

Table 6.12: Optimal thresholds found for each metric on the 'turbulent imagenet' subset, and the resulting Balanced Accuracy for both E$\overline{\text{D}}$VR and DATUM models.(Best results in **bold**, second best underlined).

	Metric	E$\overline{\text{D}}$VR		DATUM	
		Threshold	BAcc	Threshold	BAcc
NR	ARNIQA ↑	0.674	51.66%	0.508	52.52%
	BRISQUE ↓	27.629	*52.60%	45.655	52.74%
	CLIP_IQA ↑	0.631	53.17%	0.568	53.33%
	LIQE ↑	2.878	52.99%	1.134	53.72%
	MANIQA ↑	0.411	52.53%	0.236	51.79%
	NIQE ↓	4.597	51.83%	6.346	*51.68%
	PI ↓	3.922	52.82%	6.578	53.09%
	PIQUE ↓	34.821	*51.72%	53.333	51.60%
	Q-ALIGN ↑	2.283	**53.86%**	1.920	**57.20%**
	QualiCLIP ↑	0.534	52.83%	0.294	53.32%
	TOPIQ_NR ↑	0.668	53.24%	0.286	53.81%
FR	AHIQ ↑	0.424	*52.00%	0.351	*53.84%
	CW-SSIM ↑	0.994	52.90%	0.948	57.29%
	DISTS ↓	0.082	55.15%	0.162	55.77%
	FSIM ↑	0.864	*51.76%	0.904	*52.67%
	LPIPS ↓	0.102	53.07%	0.171	52.23%
	MSSSIM ↑	0.844	51.23%	0.934	51.91%
	PieAPP ↓	0.420	51.01%	1.752	*51.56%
	PSNR ↑	20.548	*51.99%	24.479	51.39%
	SSIM ↑	0.496	51.96%	0.628	51.11%
	TOPIQ_FR ↑	0.820	53.19%	0.675	53.36%
	TQI ↑	0.791	**55.62%**	0.732	**58.99%**

BAcc: Balanced Accuracy
Values marked with an asterisk (*) indicate a flip in polarity

ple, if the median of the Q-ALIGN thresholds was used (2.1015), the resulting BAcc scores would be 51.95% and 54.67% for E$\overline{\text{D}}$VR and DATUM respectively. This therefore shows that, even though the Q-ALIGN, DISTS and CW-SSIM metrics provide the best Balanced Accuracies, their use as a prediction measure of classification accuracy is limited without prior knowledge of the model quality. It is therefore recommended, if possible, that when a new model is being utilised, it is first tested with known data such that an optimal threshold can be identified for that particular model.

The second key takeaway is the simple matter of low accuracy. The best Balanced Accuracy is that of 58.99% for *TQI* on the DATUM data which, when compared to the fact that a score of 50% constitutes a 50/50 chance of correct prediction, only provides a small improvement in confidence. This therefore indicates that whilst, *on average*, metrics like Q-ALIGN or TQI can indicate if an image is classifiable, they are unable to do so with enough confidence that they can be fully relied upon. This is not the desired outcome, as the purpose of the work presented in this chapter was to find a metric that was able to potentially be used as a 'proxy' when classification was unavailable. Given these results, it is therefore difficult to recommend this use case for any of the metrics tested, as their reported accuracy is below a trustworthy range of confidence. However, these results do further prove that the use of metrics to judge image quality is an extremely challenging task, and must be approached with caution. They key factor to take into consideration is what is the purpose of the image processing task, i.e. what does 'good image quality' mean. If the ability to successfully classify an image is the goal, then the best metric to use is that of classification itself.

As for the Turbulence Text data thresholds, Table 6.13 contains the resulting values found for the NR metrics. Due to the similar behaviour of the three text classifiers, the metric values were only compared against the text classification accuracy of CRNN.

Table 6.13: Optimal thresholds found for each NR metric on the Turbulence Text dataset, and the resulting Balanced Accuracy for both $\text{E}\overline{\text{D}}\text{VR}$ and DATUM models. (Best results in **bold**, second best underlined)/

Metric	**$\text{E}\overline{\text{D}}\text{VR}$**		**DATUM**	
	Threshold	**BAcc**	**Threshold**	**BAcc**
ARNIQA ↑	0.266	*67.03%	0.398	70.34%
BRISQUE ↓	43.481	62.90%	43.458	*60.29%
CLIP_IQA ↑	0.232	*76.41%	0.196	61.43%
LIQE ↑	1.009	*70.60%	1.031	74.90%
MANIQA ↑	0.395	*63.72%	0.339	77.07%
NIQE ↓	16.480	60.08%	8.308	*64.59%
PI ↓	9.355	64.86%	6.340	*58.60%
PIQUE ↓	61.543	60.16%	45.076	<u>77.48%</u>
Q-ALIGN ↑	1.711	74.36%	1.959	65.27%
QualiCLIP ↑	0.148	***80.45%**	0.230	59.81%
TOPIQ_NR ↑	0.371	<u>*80.29%</u>	0.307	72.69%
TQI_T ↑	0.342	74.36%	0.339	**77.53%**

BAcc: Balanced Accuracy
Values marked with an asterisk (*) indicate a flip in polarity

As with the 'turbulent imagenet' results, the BAcc scores also indicate which of the two models is more consistent. The highest BAcc for DATUM is 77.53%

compared to 80.45% for $\text{E}\overline{\text{D}}\text{VR}$, indicating that, in the case of the Turbulence Text Dataset, the DATUM model is able to produce higher quality images, resulting in a more difficult task for the metrics. Unlike the 'turbulent imagenet' results, in the case of the text data, the metrics are far more effective at differentiating between classifiable and unclassifiable images, with the highest accuracy being obtained by the *inverse* of QualiCLIP. This shows that QualiCLIP would not be useful as a measure for good quality images, but rather a measure for which images are too 'noisy' for successful classification. Of the metrics that do not need polarity inversion, it is again Q-ALIGN and TQI_T that perform the best for the $\text{E}\overline{\text{D}}\text{VR}$ outputs, and TQI_T for DATUM. It is the latter of these that is most interesting, as the TQI_T metric is dependent upon the scores from MANIQA. However, after TQI_T, it is PIQUE that obtains the next best BAcc accuracy. This is interesting, as it would be expected that, due to the nature of TQI_T, its BAcc would be closer to that of MANIQA. This therefore shows the added benefit of incorporating Q-ALIGN into the TQI_T metric as, when combined with MANIQA, it is able to outperform both. Similarly, BRISQUE shows very poor performance when compared to its correlation ranking. However, as mentioned previously, the range of the BRISQUE scores is extremely low, which is not factored into the correlation scores. This does however result in poor performance when choosing a suitable threshold. This is also shown by the correlation results (for the CRNN classifier) the only metrics that do not change their opinion on image quality between $\text{E}\overline{\text{D}}\text{VR}$ and DATUM are TQI_T, PIQUE and Q-ALIGN, where Q-ALIGN and TQI_T are the more consistent of the three. These results are far more promising than those found for 'turbulent imagenet', in that the metrics are able to provide a high degree of confidence that the images are either classifiable or not. It is therefore easier to recommend their use in the case of images similar to those found in the Turbulence Text dataset. Given its consistency between models in both correlation and BAcc scores, TQI_T is the recommended metric to use as a prediction of successful text classification.

6.6 Discussion

This study provided a great deal of information regarding the use of metrics for turbulence mitigated image quality. Firstly, the results in this chapter show that the measurement of image quality is an extremely difficult task, and is dependent on multiple factors, such as the model being used and the data being processed. It has been shown that no single metric is able to provide a clear assessment of image quality for all cases. Q-ALIGN and DISTS were determined to be the best performing original metrics for NR and FR respectively, however an over reliance on just these two would be inadvisable. The two new metrics introduced, TQI and TQI_T, aim to overcome this by incorporating several metrics into their evaluation, and therefore provide a more rounded quality score. Although showing good performance, the metrics were not able to fully distinguish between classifiable and unclassifiable images, especially in the case of 'turbulent imagenet' where, if using TQI as a prediction of classification accuracy, a probability of 55.62% or 58.99% would not be significant enough in order to recommend such a metric. It is therefore encouraged that if available, the metric for turbulence mitigation should be that of the post-processing task.

What these results do provide is a comparison between the metrics. This therefore allows the choice of metrics to be approached with a better understanding, as well as an understanding of which metrics would work *together*. As highlighted in [172], the most efficient method of determining image quality is by making use of as many metrics as possible in order to gain a full understanding. By knowing the behaviour of each metric to different types of distortion, the full 'quality' of an image can be inferred. In the case of the Turbulence Text dataset for example, due to the nature of the TQI_T metric, it is able to use the MANIQA score as a 'noise indicator', and therefore apply the best metric to each situation. Another use could be that of the QualiCLIP metric, which presented a the highest *negative* correlation. This, used alongside a positively correlated metric (such as Q-ALIGN), could be used to understand the image contents. If QualiCLIP deems the image as 'good' quality, then it is actually less likely to be classified. If, at the same time, the corresponding Q-ALIGN score deems the quality as 'low', then this assessment is corroborated.

The results also reveal that the threshold of the metrics are very sensitive to the data quality. Take the 'turbulent imagenet' outputs from the E$\overline{\text{D}}$VR model for example. On the whole, the image quality is higher, therefore the metric thresholds are less able to distinguish between classifiable and unclassifiable images. If that same threshold were to be used for a lower quality model, such as DATUM, it would result in a significant drop in accuracy, implying that the 'one-shot' performance of a given threshold is very poor. This therefore provides a warning to the use of such metrics for classification accuracy prediction, and promotes the use of calibration before application. If a new model is being utilised, before applying metrics, a new 'optimal' threshold should be evaluated with known data. This allows more confidence in the use of such a threshold on unseen data.

A final highlight of these results is the fact that previous literature has placed too large a reliance on the optimisation of non-optimal metrics such as PSNR and SSIM. The results presented show that, by aiming for high values of such metrics, the resulting images are actually of limited in their use for real world applications. The issue lies however with the fact that the over-reliance on such metrics further promotes their use. To outperform previous architectures or achieve state-of-the-art status, these benchmarks are the most common targets and are therefore upheld as the optimal goal. It is argued that use of these metrics is leading the field of turbulence mitigation down the wrong path, where novel research is 'aiming for the wrong target'. If more metrics were to be used on a regular basis, the comparison of different algorithms would be far richer, therefore allowing a true progression of turbulence mitigation algorithms.

6.7 Conclusion

This chapter proposed the potential misuse of common metrics such as PSNR and SSIM for the specific case of turbulence mitigated image quality assessment. These metrics were shown to contradict the perceived perceptual quality of images, and it was argued that current literature is over-reliant on such metrics. This therefore creates an arena of competition that is aiming to maximise a metric that is unsuitable for real life applications

Given this, it was determined that a study of available image quality metrics for the specific task of turbulence mitigated image quality assessment would be highly desirable. A suite of 21 metrics were chosen that spanned both NR and FR metrics, and contained traditional commonly used metrics as well as modern deep learning approaches.

In order to first support the argument presented, a survey was conducted, where participants were shown two images and asked to select the one that they believed to be of 'better' quality. When these ratings were compared to the metrics, it was proven that the commonly used metrics of PSNR and SSIM disagree with that of perceptual quality.

Before applying the metrics to the task of turbulence mitigation, it was first deemed that an understanding of each metric's response to common distortions would be useful. Therefore, a dataset was synthesised containing eight different distortions applied to five different source datasets. Using this data, the response of each metric was obtained and analysed. These results showed that for the stepped levels of each distortion, each metric was successful in determining the decrease in image quality, where the FR metrics were found to perform the best. Other useful insights gained were the fact that NR metrics are extremely poor at identifying saturation differences, and that the CW-SSIM metric had the best performance for turbulent data.

The levels of the distortions however presented clear differences in image quality. When performing turbulence mitigation, the output images represent a far closer distribution of quality, in that the difference in quality is harder to distinguish. The next step of the study was to therefore analyse how the metrics reacted when provided with turbulence mitigated examples. These examples had first been analysed and identified with perceived distortions such as blur and noise. With these perceived distortions known, the metric response was analysed to understand their correlation with human perception. It was found that some metrics successfully aligned with the perceived perception, whilst others did not perform as well. This suggests that some of the metrics may not prioritise the same features that humans associate with high-quality images.

Finally, it was proposed that what a human perceives as high-quality may not be the best metric either. As in many modern scenarios, post turbulence mitigation tasks such as tracking or classification are commonly done autonomously. It is therefore these algorithms that the turbulence mitigation networks should be tailored towards. Therefore, with the use of the 'turbulent imagenet' dataset presented in Chapter 4 alongside the Turbulence Text dataset [60], image and text classification results could be obtained as a post turbulence mitigation step. These classifications could then be compared against the metrics to identify which, if any, correlated with the classification results. It was found that some metrics did indeed provide positive correlation, however, for the case of the data from 'turbulent imagenet', this correlation was very slight, and was only marginally better than a 50/50 random chance. For the text data, the metric response was more promising where, interestingly, it was found that the best performing metric actually presented *negative* correlation, but could be used in reverse to provide a classification accuracy prediction. Using the results from the correlation investigation, two new metrics were proposed: TQI and TQI_T. These aimed to bridge the gap between different metrics by applying several at once.

The resulting outcomes of this chapter are comprehensive. The over-reliance on common metrics such as PSNR and SSIM was first proven, and an in depth study of potential replacement metrics was undertaken. Using these results, future research can be undertaken with a better understanding of what is important for turbulence mitigation, and which metrics will help with that goal.

Chapter 7

Conclusions

Atmospheric turbulence induces light distortions that adversely affect the quality of both images and video, therefore posing challenges in tasks such as classification and tracking. It is therefore desirable to mitigate these distortions such that downstream tasks can be optimised. This thesis has presented an investigation of using deep learning for this task, and highlighted the various challenges involved. The following are the key findings of this thesis.

Development of a turbulence simulation tool: Due to a lack of turbulence data in literature, and the complexities regarding real-life data acquisition, the development of a turbulence simulator was first necessary to facilitate the a use of deep learning for turbulence mitigation. The resulting simulator developed in this thesis is capable of processing both single images as well as video sequences. It also allows a high level of control over the resulting distortions via comprehensive input parameters. The simulator was validated against theoretical expectations, and was confirmed to represent a true turbulent atmosphere. With this simulator, various datasets were synthesised and used to train and test algorithms in Chapters 5 and 6. Of these datasets 'turbulence videos' has been made publicly available.

Identification of optimal deep learning strategies for deep learning: When this research was begun, the application of deep learning to turbulence mitigation was not widely reported. It was therefore not yet clear which deep learning architecture was best suited to the task. Therefore, to identify such an architecture, a series of off-the-shelf models were obtained and re-trained on the turbulence data generated in Chapter 4, where it was found that the underlying distortions could indeed be mitigated via deep learning, and that some architectures performed better than others. This investigation prompted the use of turbulent videos, as seen in traditional approaches, where the use of temporal data allows a network to gain a better representation of the underlying latent image. With this in mind, the EDVR architecture was utilised and trained on turbulence video sequences, as well as an additional perceptual loss, with the aim to improve the accuracy of post-mitigation classification. This same concept was then applied to the state-of-the-art model DATUM, where it was indeed found that the optimisation of the loss function leads to better mitigation performance. The key finding from this investigation is that effective deep learning–based turbulence mitigation depends not only on the choice of architecture, but also critically on the exploitation of temporal information and the optimisation of the loss function. Architectures that incorporate video sequences consistently outperform single-frame models, and perceptual-loss–based training reliably enhances downstream classification accuracy without requiring structural modifications.

Analysis of image quality metrics for turbulence mitigation: During this research, it was found that metrics such as PSNR and SSIM were widely used in image processing applications. This was also true for the field of turbulence mitigation. However, via visual analysis, it was found that these metrics may have been providing misleading information regarding the true quality of the output images. Given this, an investigation was undertaken to identify which metrics are best suited for the quality assessment of turbulence mitigation algorithms. To achieve this, the most likely use of turbulence mitigation was used as a control, that being post-mitigation classification. By comparing reported quality with classification accuracy, the metrics that best correlate were found. Furthermore, these results led to the development of two new image quality metrics, TQI and TQI_T. In this, it was reported that TQI was the optimal full-reference metric, and TQI_T was the optimal no-reference metric. It was also put forward that, if classification is the underlying motivation for turbulence mitigation, then it should be classification that is used as the quality metric.

In conclusion, this thesis has presented an investigation into the practical use of deep learning for turbulence mitigation. It has shown the capability of data-driven algorithms to successfully capture the underlying distortions caused by a turbulent atmosphere and recover a clean image. Since the largest issue facing deep learning progression is that of data availability, this thesis has also presented a turbulence simulator that is able to apply the effects of turbulence onto any image. The assessment of mitigation quality was also investigated, where it was found that commonly used image quality metrics are not suitable for turbulence mitigation. Other metrics were therefore tested, and compared with classification accuracy, to identify which metric would be optimal for turbulence mitigation quality assessment. From this thesis, the field of turbulence mitigation can extract a great deal of useful information that would be invaluable for future work.

7.1 Future Work

Given the work presented in this thesis, there are a number of possible future directions that can this research can be taken. These are presented here.

Expansion of Simulation Capability. Chapter 4 described the development of an accurate turbulence simulation tool. Further improvements to this simulation could include the incorporation of RGB simulation, or expansion to dynamic data, whereby the simulator would be capable of applying turbulence distortions onto an existing video sequence, as opposed to creating frames from a single input image. Finally would be the simple expansion of publicly available datasets by synthesising new turbulent sequences.

Additional experiments into perceptual loss. The implementation of the perceptual loss in Chapter 5 was done in line with previous works, and made use of only one feature layer of the VGG model. Future experiments could build on this work by extracting features from different layers, combining multiple layers, or even using an alternative classification model.

Leverage Metric Investigation Results to Inform Deep Learning. The findings in Chapter 5 showed that the loss function is a key factor in any deep learning approach. Therefore, a potential future avenue of research is the leveraging of results from Chapter 6, where it was shown that some metrics are better suited towards turbulence mitigation. By applying the concepts from these preferred metrics, a bespoke loss function could be developed that would greatly increase the performance of deep learning turbulence mitigation techniques.

Further investigation into optimised image quality metric. Presented in Chapter 6 are two new metrics for turbulence mitigation model analysis: TQI and TQI_T. These metrics utilise the quality scores from other metrics to inform a final quality decision, a decision that is made via simple combination or choice of the existing scores. A more complex metric may, instead of simply combining other metrics, feed them into a more complex algorithm in order to identify a final image score. For example, instead of the equation for TQI as shown, the three metrics in question could be fed into an ANN, which can be trained to identify patterns within the results to provide a more nuanced image quality score.

Multi Domain Analyse of Image Quality Metrics. The use of classification in Chapter 6 allowed the identification of metrics that best correlated with classification accuracy. However, these metrics may not necessarily retain their usefulness in other domains, such as tracking or segmentation. A natural progression therefore to this research is the expansion to multiple application domains. Such an investigation may lead to key findings into the priorities of turbulence mitigation techniques, and may allow a bespoke turbulence metric to be developed.

Appendix A

Single Frame Deep Learning Architectures

This appendix provides the deep learning architectures used in Chapter 5 for the investigation into single frame deep learning models for turbulence mitigation.

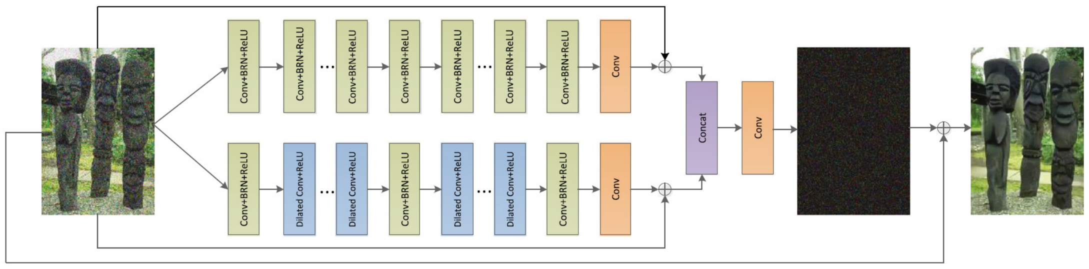
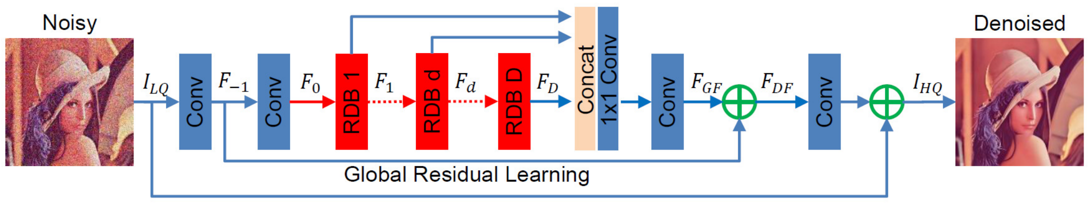
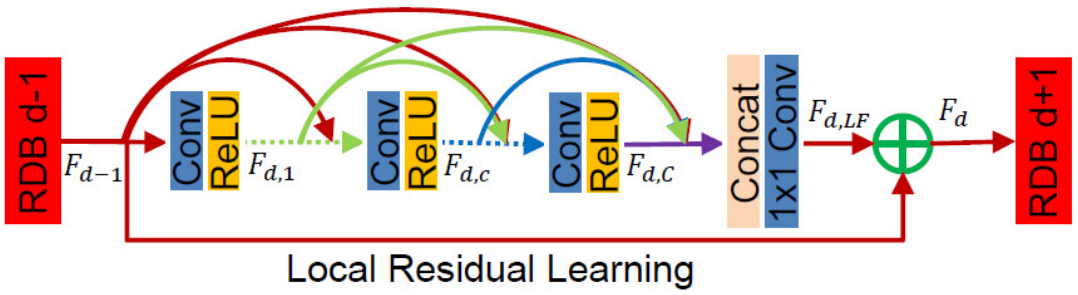


Figure A.1: The CNN architecture of the BRDNet [234]



(a) Architecture of RDN [235]



(b) Residual Dense Block used in the RDN architecture [235]

Figure A.2: Overview of the RDN architecture and its Residual Dense Block.

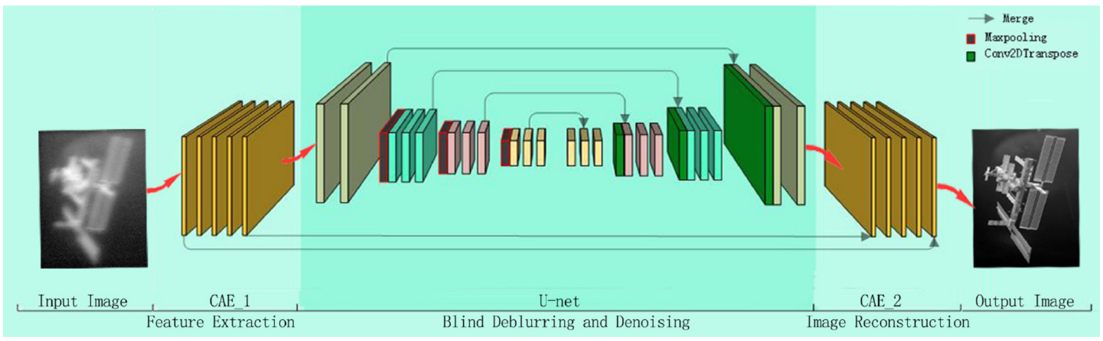


Figure A.3: Architecture of CAE-Unet-CAE [33]

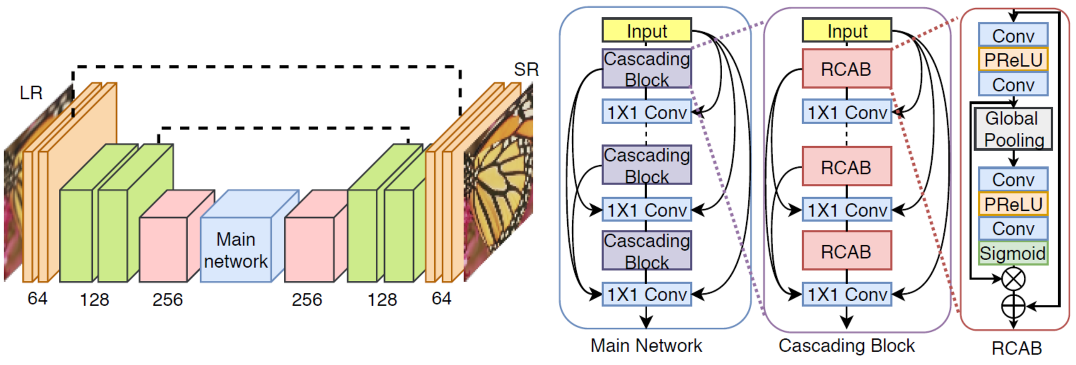


Figure A.4: Architecture of SuperSR [236]

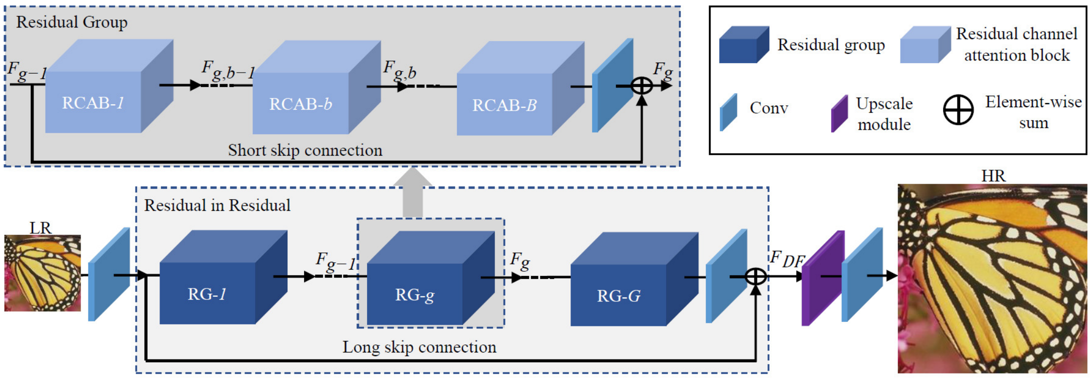


Figure A.5: Architecture of RCAN [237]

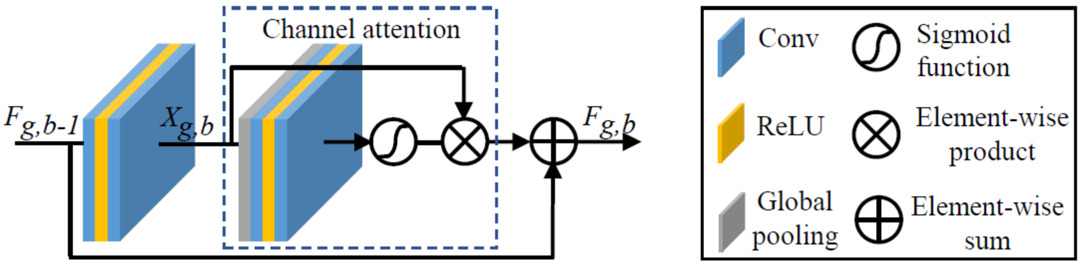


Figure A.6: Architecture of the RCAB Block used in both RCAN and SuperSR [237]

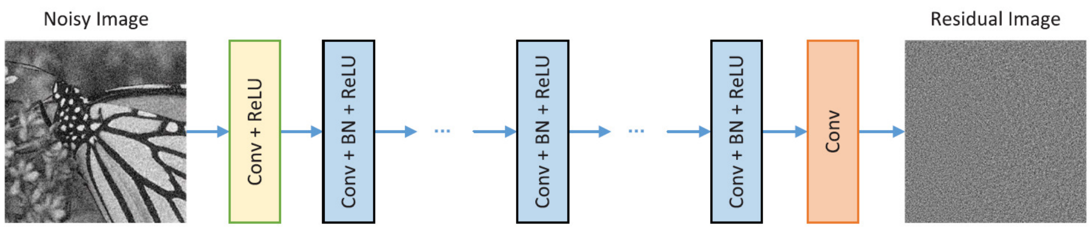


Figure A.7: Architecture of DnCNN [136]

Appendix B

Additional Simulator Outputs

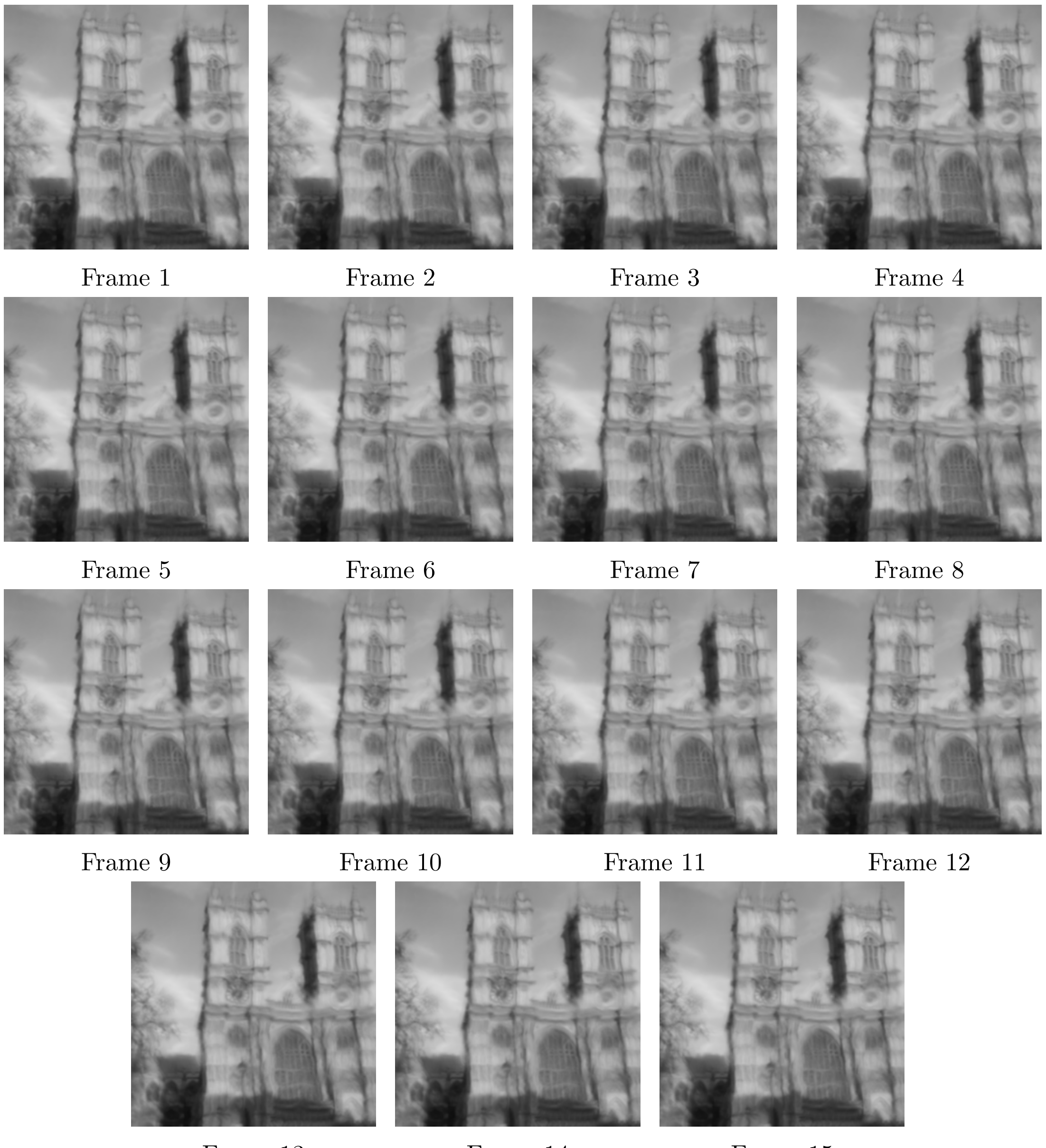


Figure B.1: Fifteen 'Slow' turbulent frames, synthesised using turbulence simulation tool. $C_n^2 = 0.875 \times 10^{-15} m^{-2/3}$, turbSpeed=1, turbDirection=$\uparrow$, $l_0 = 0.01m$, $L_0 = 300m$, $\lambda = 525nm$, Sub_Count=2, $N_{subharm} = 4$, $D = 0.2034m$, $L = 7000m$, $nscr = 10$, $skip = 4$, Image_Pixels=257

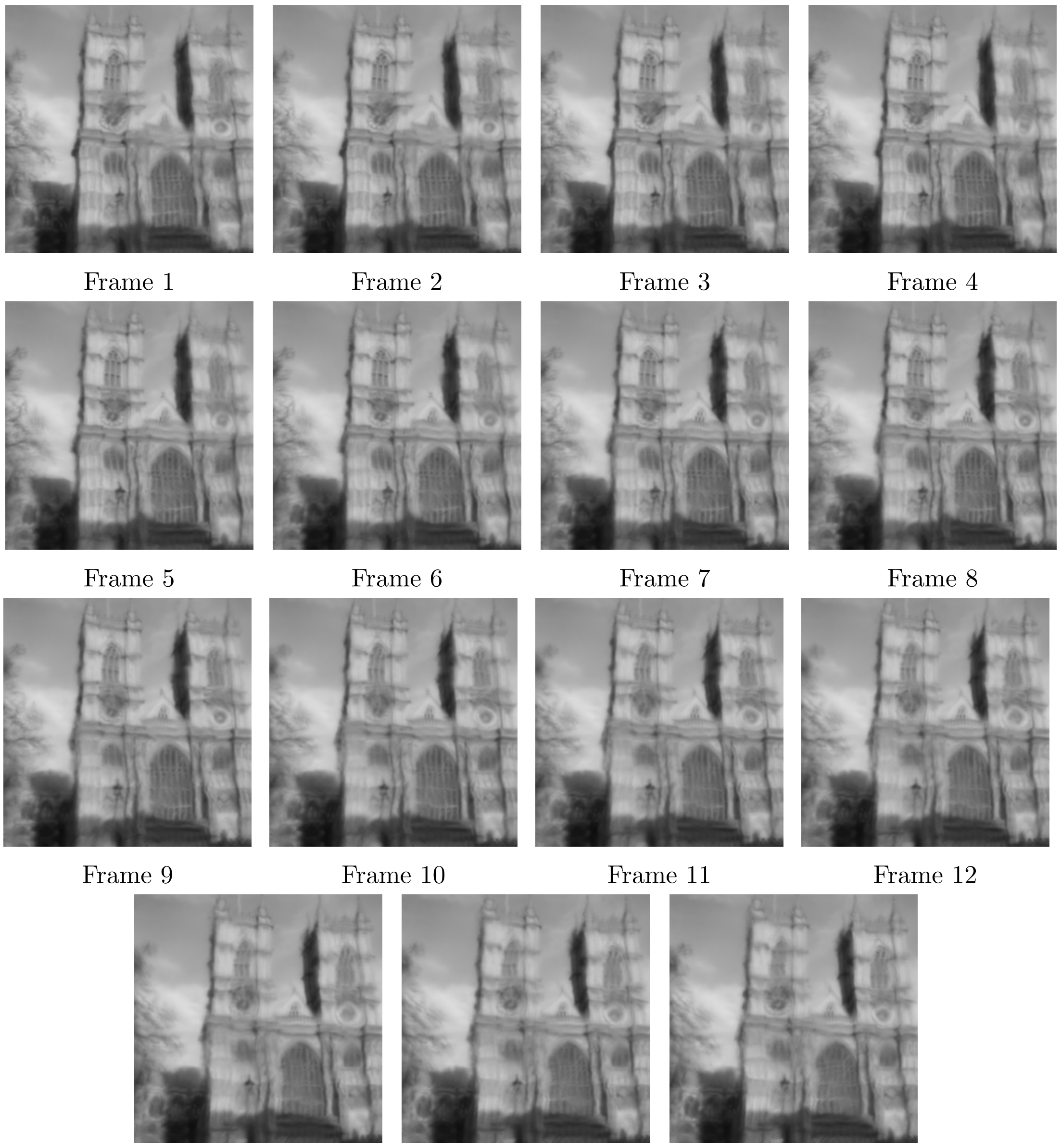


Figure B.2: Fifteen 'Fast' turbulent frames, synthesised using turbulence simulation tool. $C_n^2 = 0.875 \times 10^{-15} m^{-2/3}$, turbSpeed=4, turbDirection=$\uparrow$, $l_0 = 0.01m$, $L_0 = 300m$, $\lambda = 525nm$, Sub_Count=2, $N_{subharm} = 4$, $D = 0.2034m$, $L = 7000m$, $nscr = 10$, $skip = 4$, Image_Pixels=257

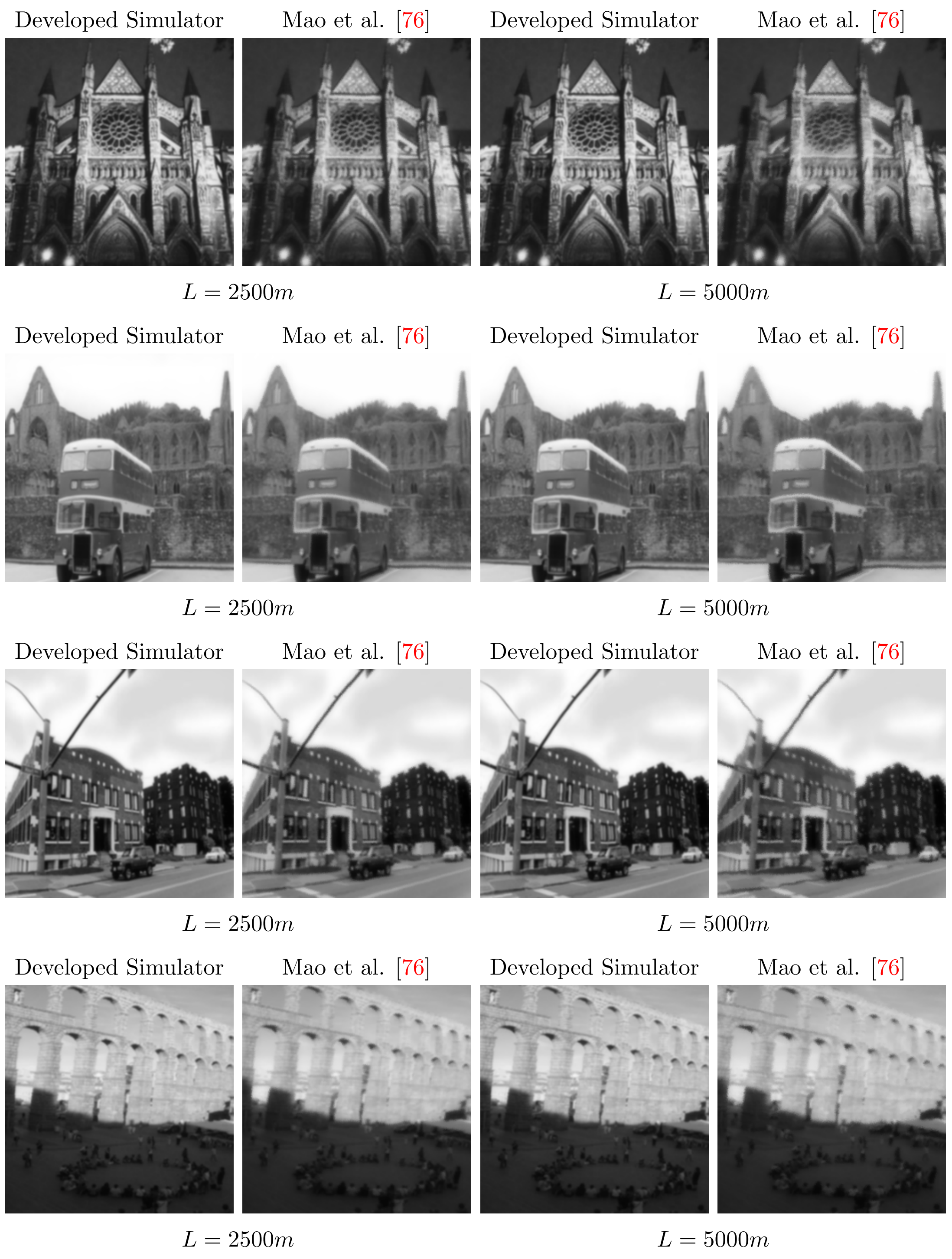


Figure B.3: Example outputs from the developed simulation tool, compared with the work from Mao et al [76]. Two outputs from each are shown: $L = 2500m$ and $L = 5000m$. Both synthesised with a turbulence strength of $C_n^2 = 0.1 \times 10^{-15} m^{-2/3}$

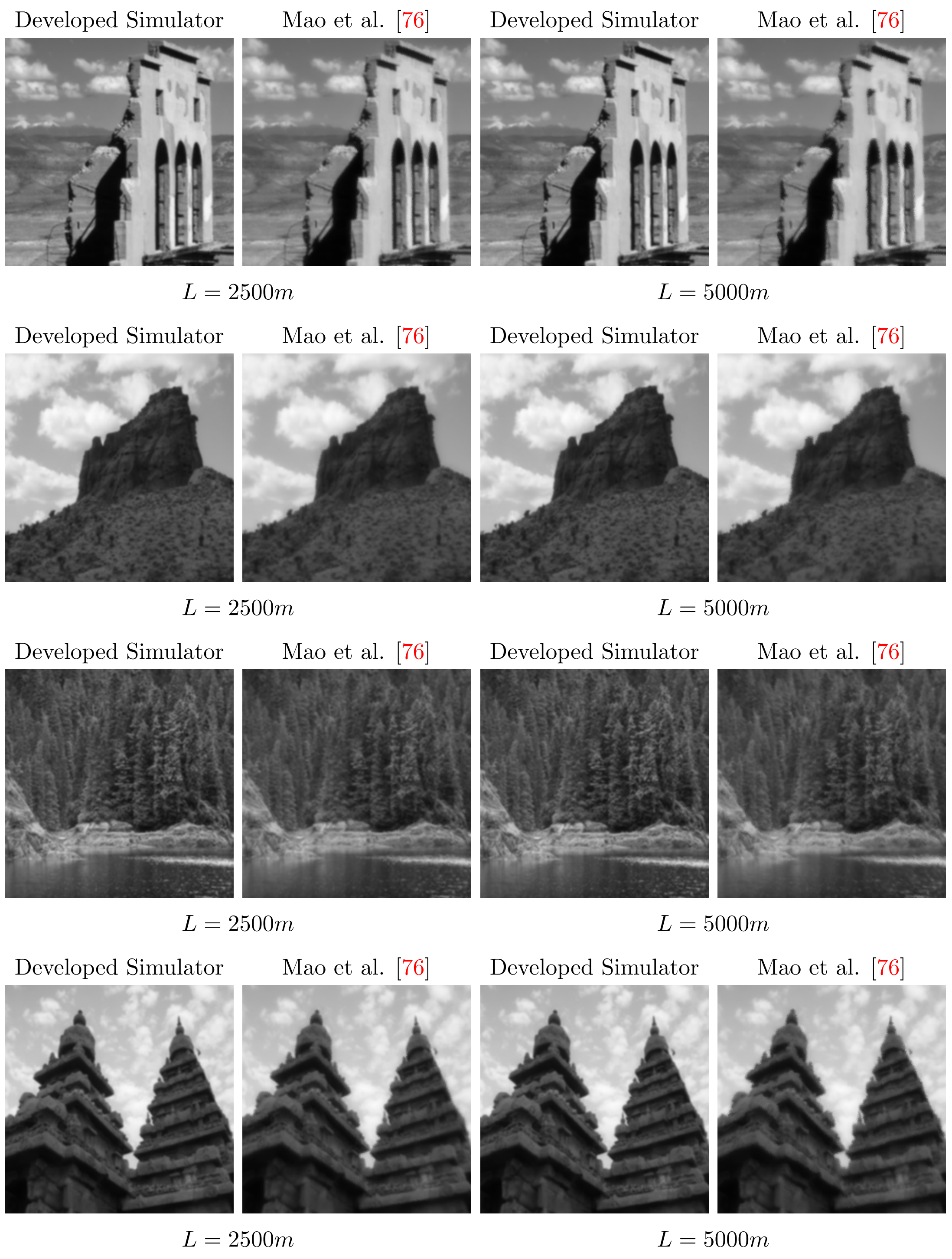


Figure B.4: Example outputs from the developed simulation tool, compared with the work from Mao et al [76]. Two outputs from each are shown: $L = 2500m$ and $L = 5000m$. Both synthesised with a turbulence strength of $C_n^2 = 0.1 \times 10^{-15} m^{-2/3}$

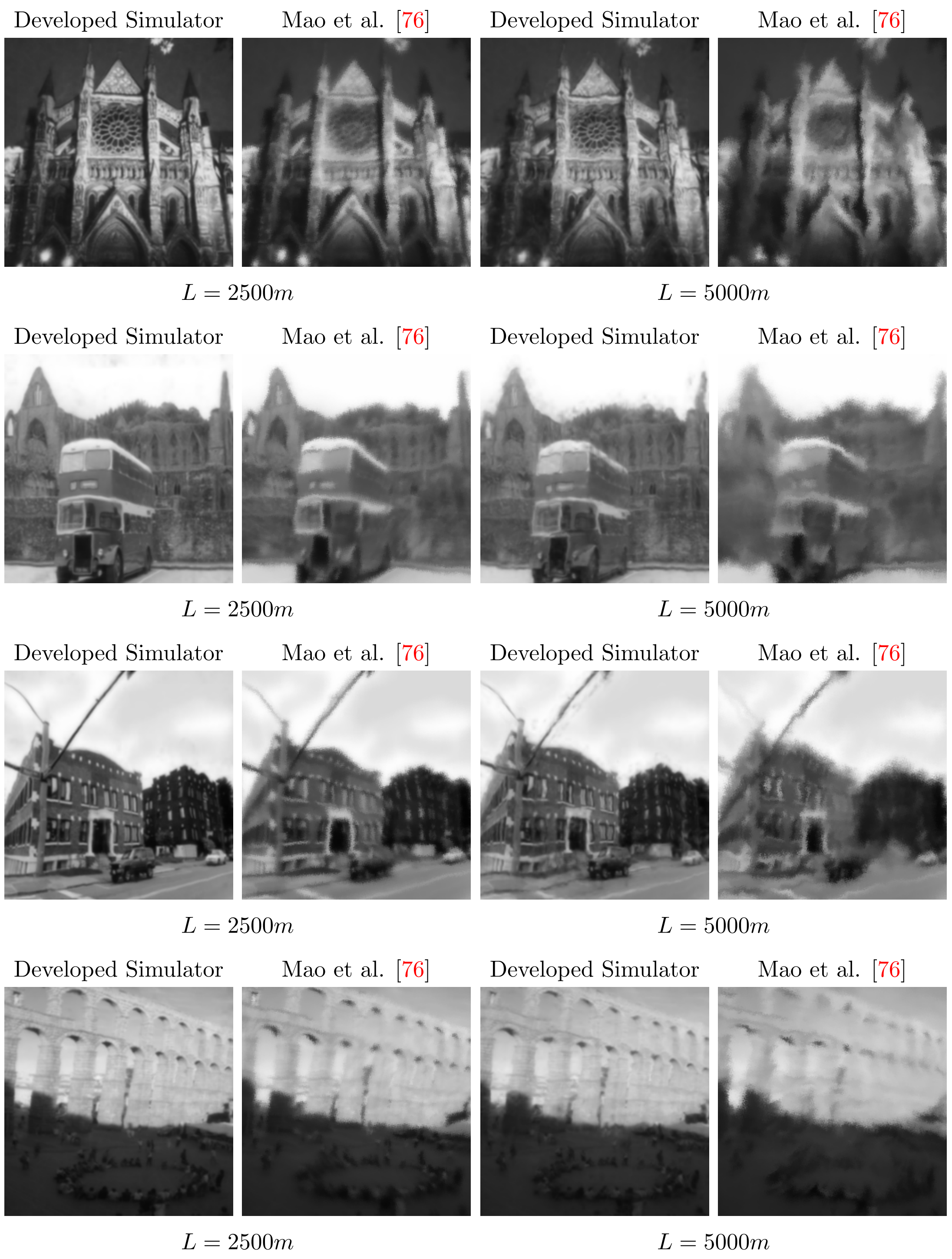


Figure B.5: Example outputs from the developed simulation tool, compared with the work from Mao et al [76]. Two outputs from each are shown: $L = 2500m$ and $L = 5000m$. Both synthesised with a turbulence strength of $C_n^2 = 1.5 \times 10^{-15} m^{-2/3}$

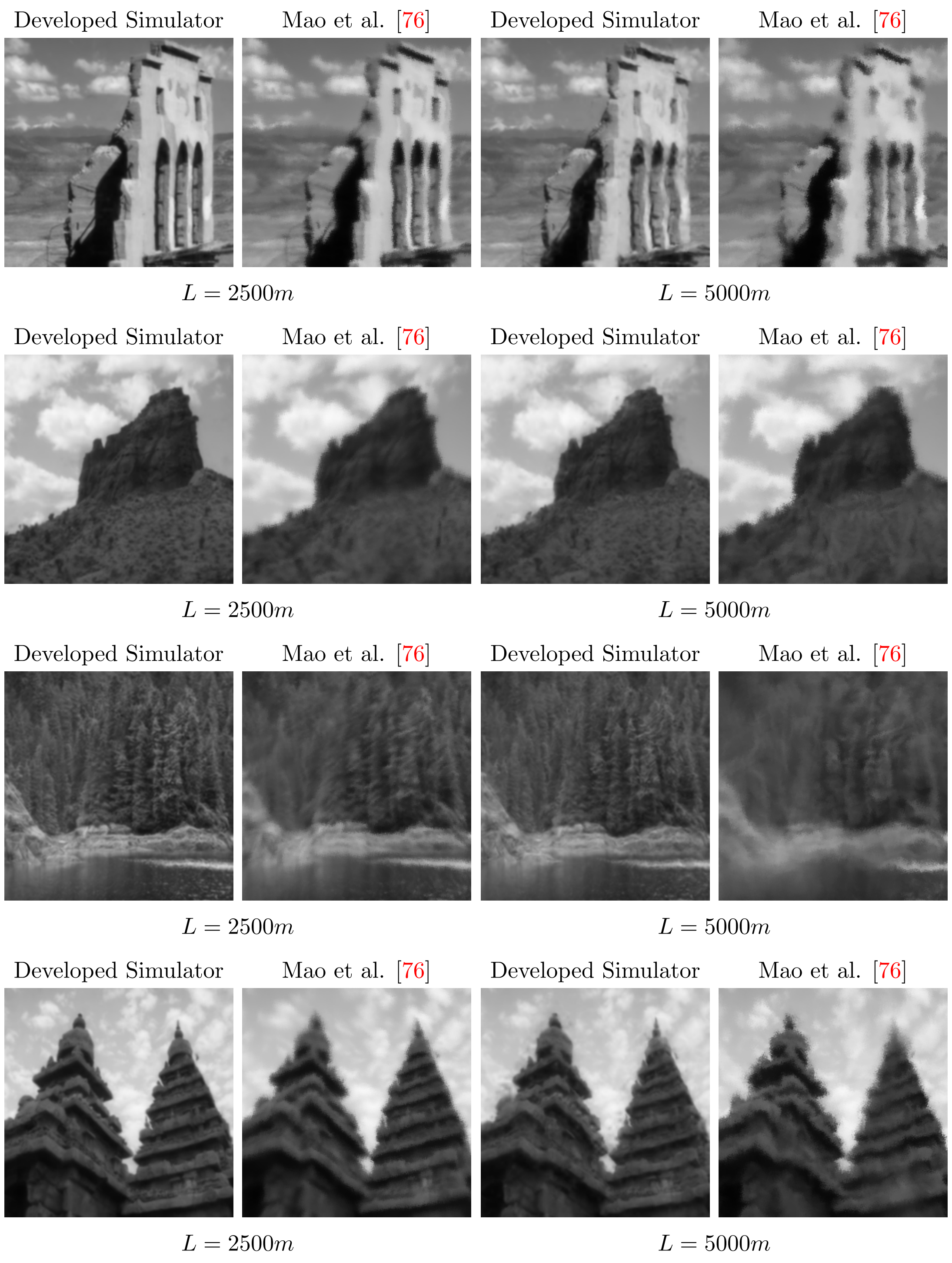


Figure B.6: Example outputs from the developed simulation tool, compared with the work from Mao et al [76]. Two outputs from each are shown: $L = 2500m$ and $L = 5000m$. Both synthesised with a turbulence strength of $C_n^2 = 1.5 \times 10^{-15} m^{-2/3}$

Appendix C

Metric Scores Compared with Classification Accuracy

Provided in this appendix are the graph metric score graphs for the two datasets: 'turbulent imagenet' and Turbulence Text dataset, which have been processed by the two models: $\text{E}\overline{\text{D}}\text{VR}_s$ and DATUM.

Firstly, the NR and FR metric results for 'turbulent imagenet' are provided, followed by NR results for the Turbulence Text dataset, where the classifications of CRNN, DAN and ASTER.

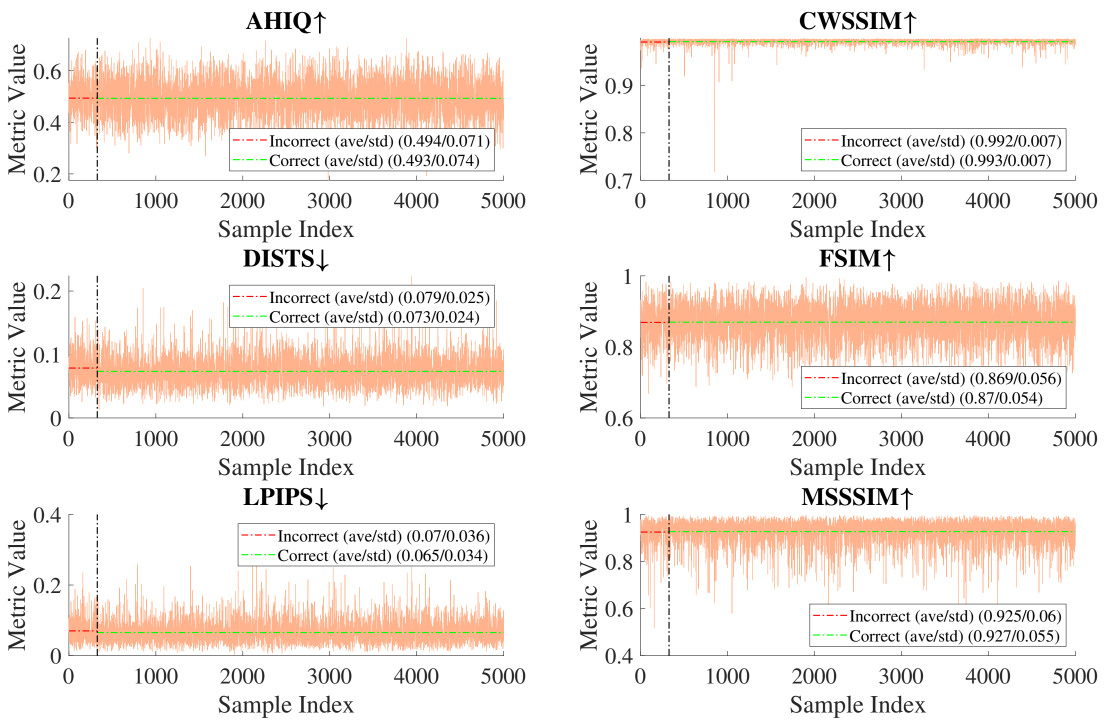


Figure C.1: FR metric scores on the 'turbulent imagenet' data processed by $\text{E}\overline{\text{D}}\text{VR}_s$ compared with the resulting classification accuracy (93.4%)

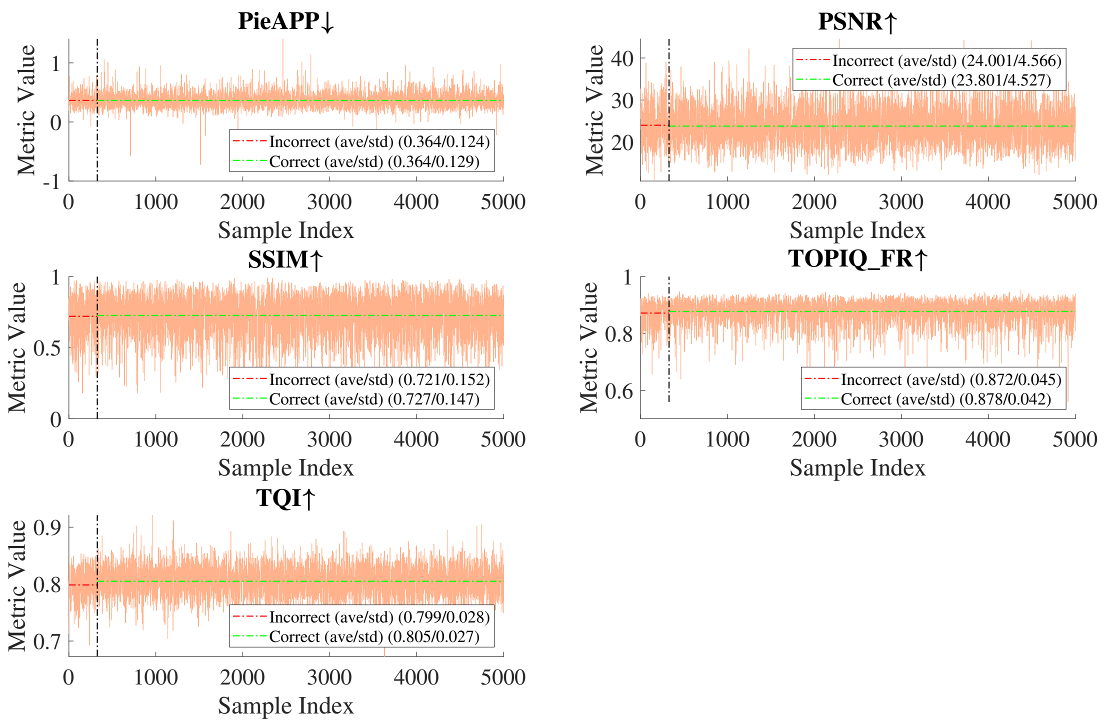


Figure C.2: FR metric scores on the 'turbulent imagenet' data processed by E$\overline{\text{D}}$VR$_s$ compared with the resulting classification accuracy (93.4%)

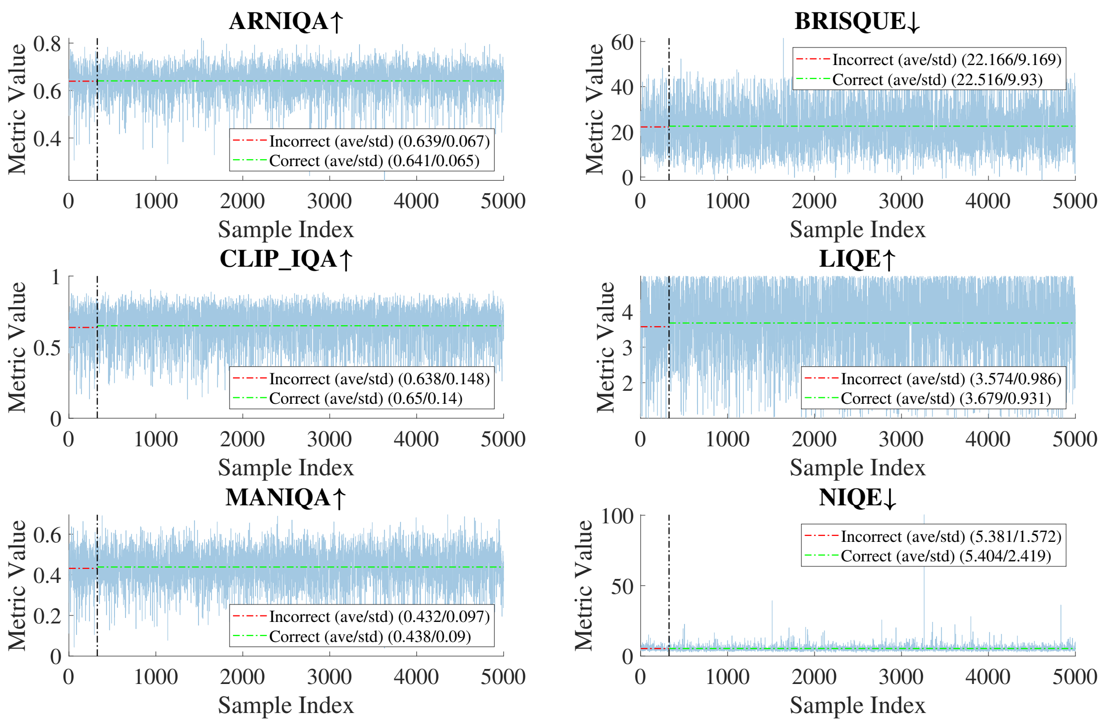


Figure C.3: NR metric scores on the 'turbulent imagenet' data processed by E$\overline{\text{D}}$VR$_s$ compared with the resulting classification accuracy (93.4%)

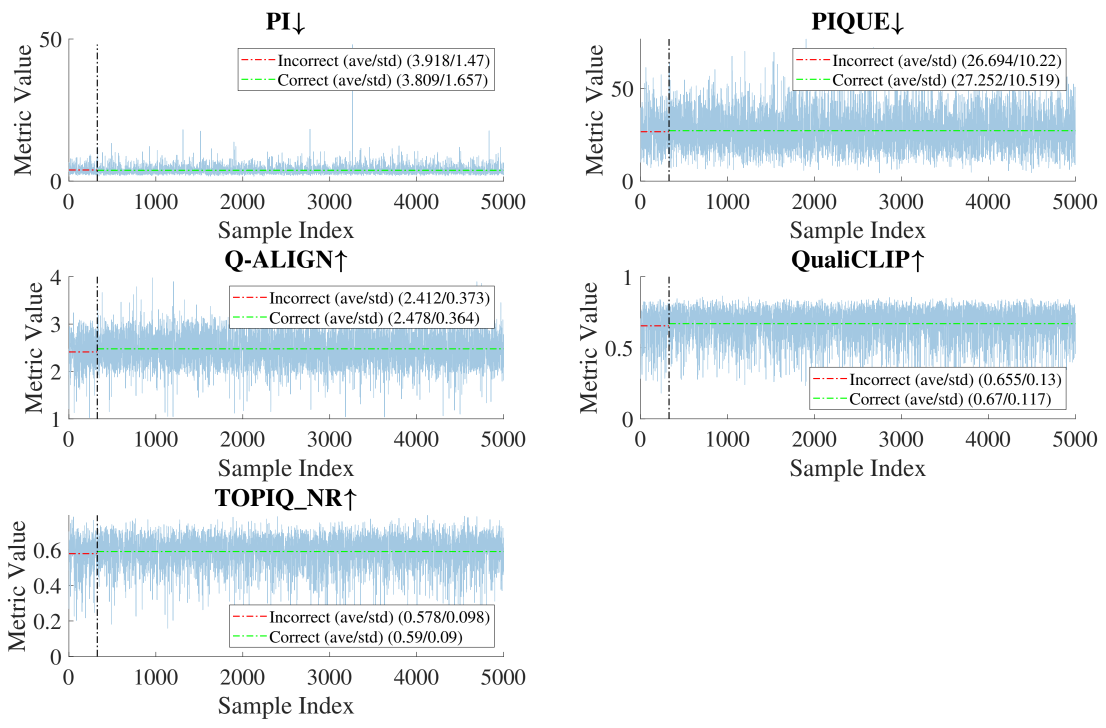


Figure C.4: NR metric scores on the 'turbulent imagenet' data processed by $\mathrm{E\overline{D}VR}_s$ compared with the resulting classification accuracy (93.4%)

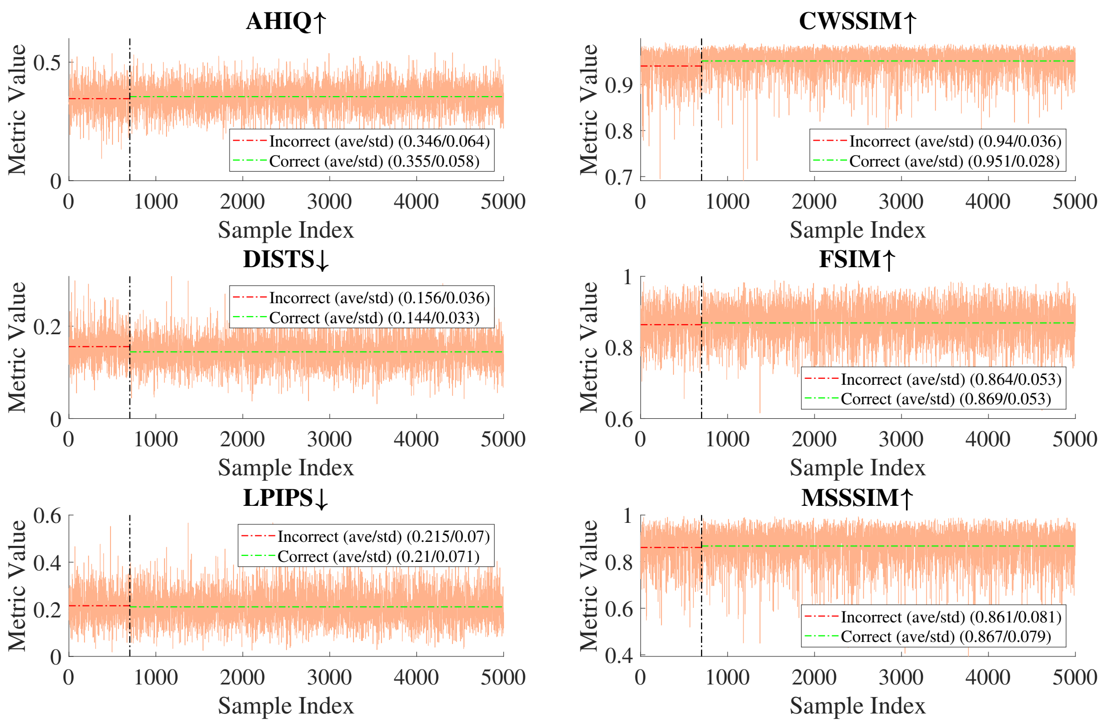


Figure C.5: FR metric scores on the 'turbulent imagenet' data processed by DATUM compared with the resulting classification accuracy (86%)

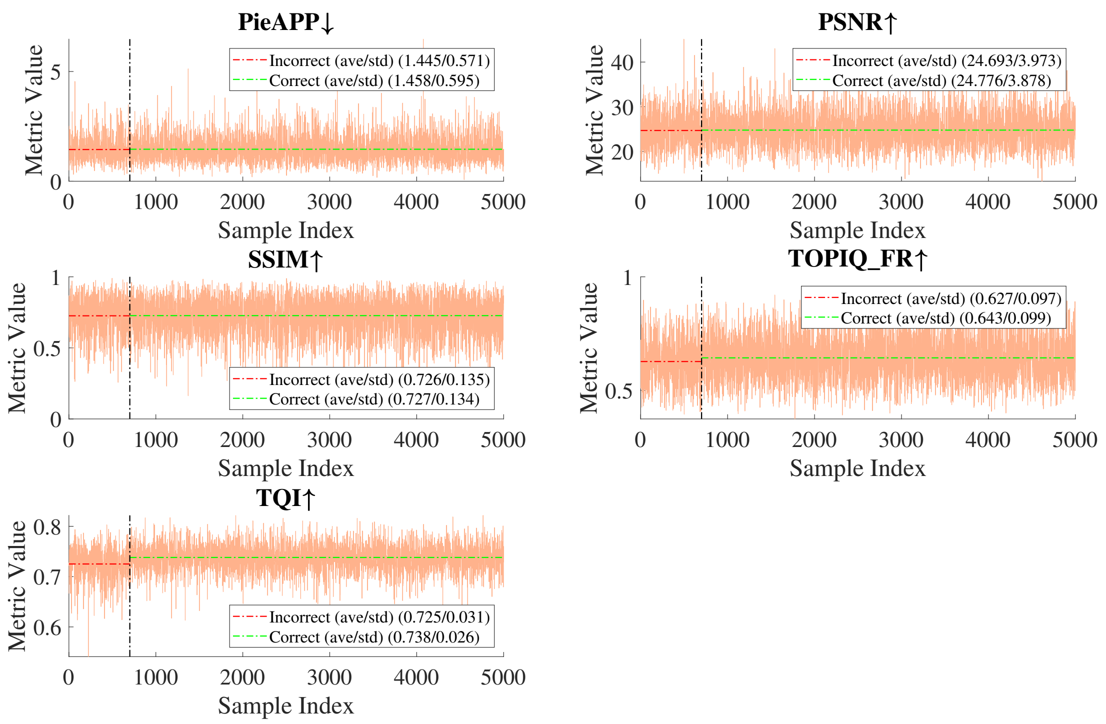


Figure C.6: FR metric scores on the 'turbulent imagenet' data processed by DATUM compared with the resulting classification accuracy (86%)

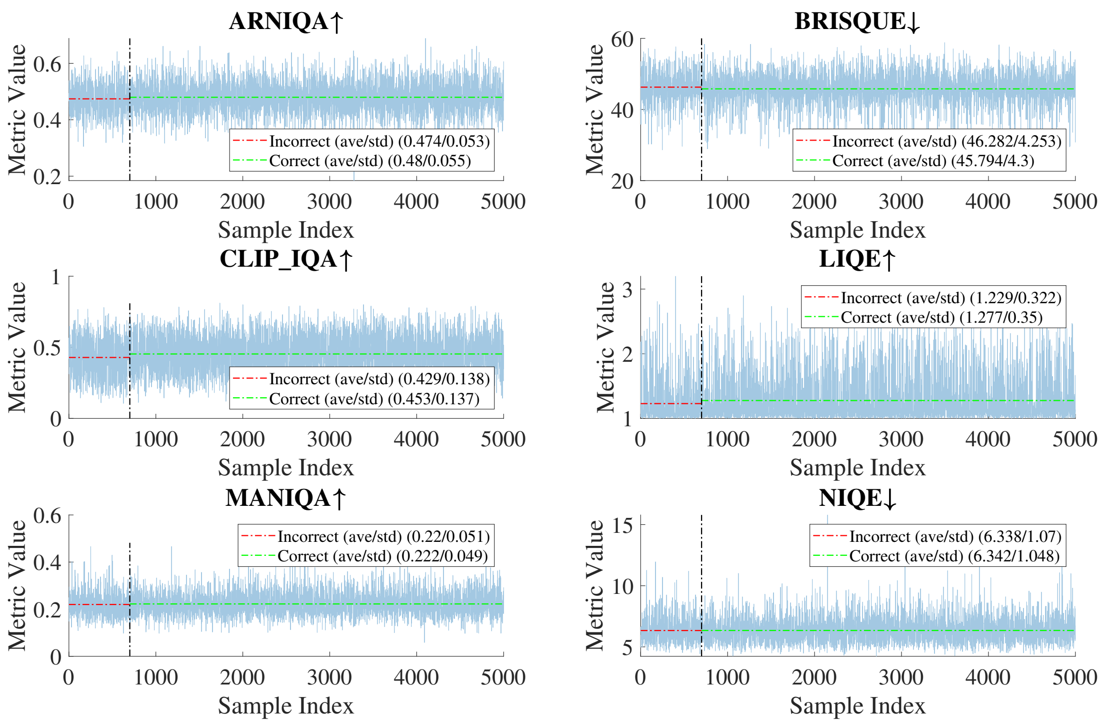


Figure C.7: NR metric scores on the 'turbulent imagenet' data processed by DATUM compared with the resulting classification accuracy (86%)

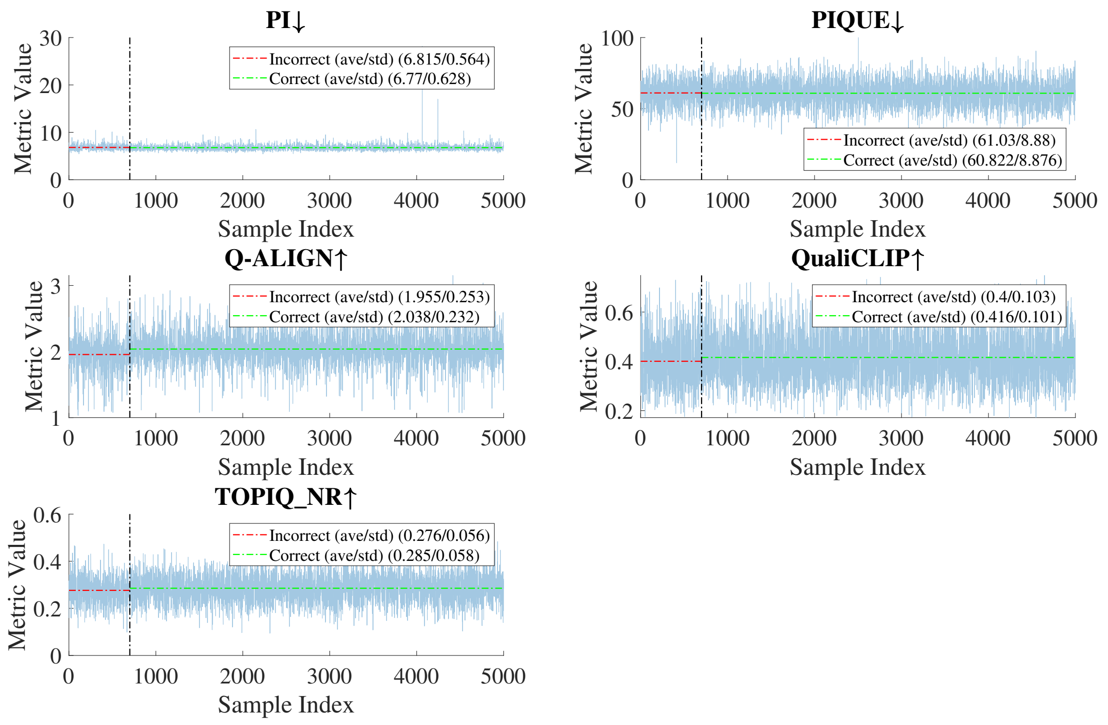


Figure C.8: NR metric scores on the 'turbulent imagenet' data processed by DATUM compared with the resulting classification accuracy (86%)

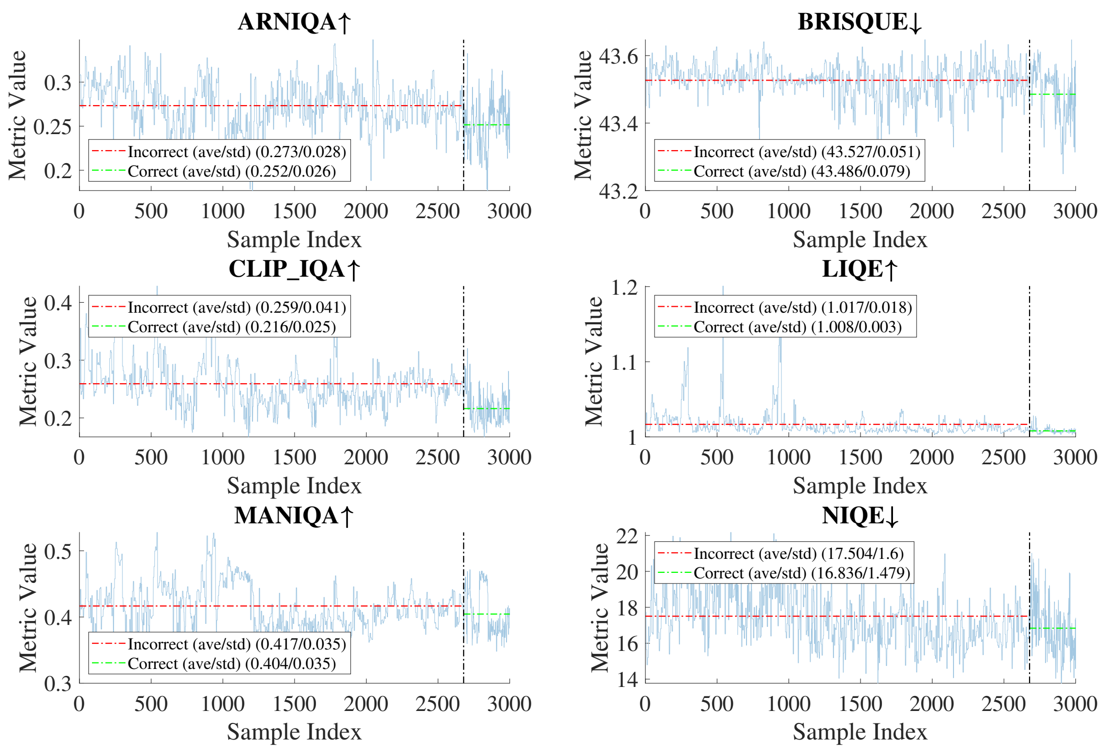


Figure C.9: NR metric scores on the Turbulence Text data processed by $\mathrm{E\overline{D}VR}_s$ compared with the resulting CRNN classification accuracy (10.73%)

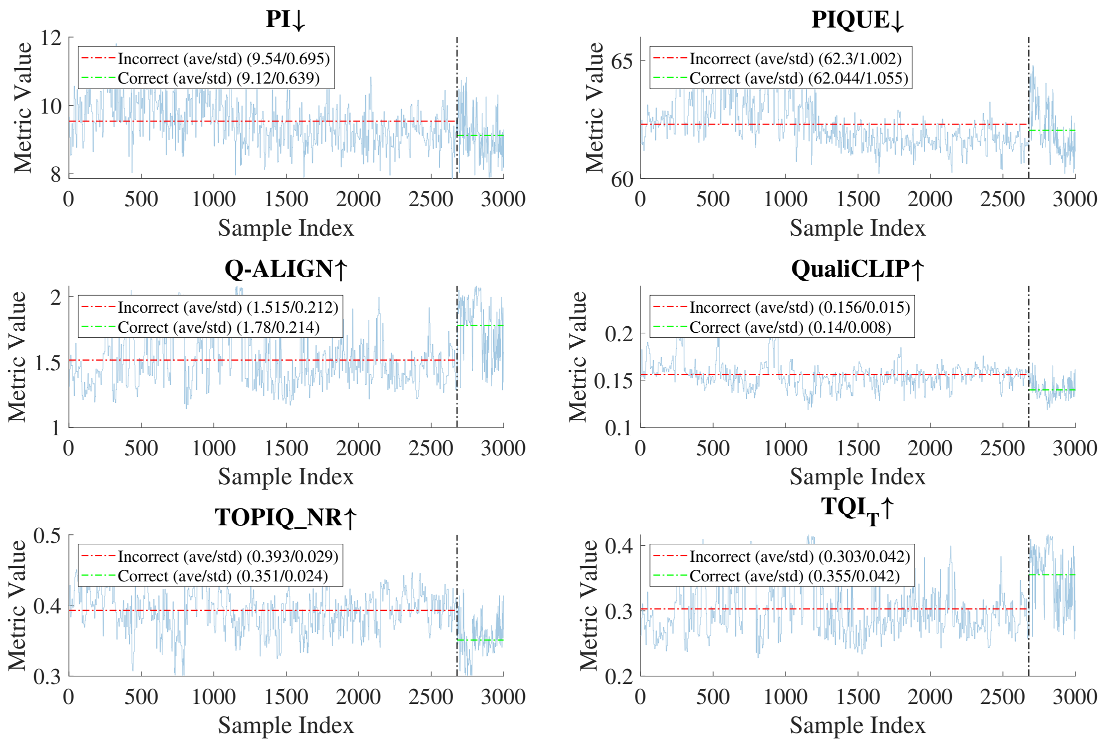


Figure C.10: NR metric scores on the Turbulence Text data processed by E$\overline{\text{D}}$VR$_s$ compared with the resulting CRNN classification accuracy (10.73%)

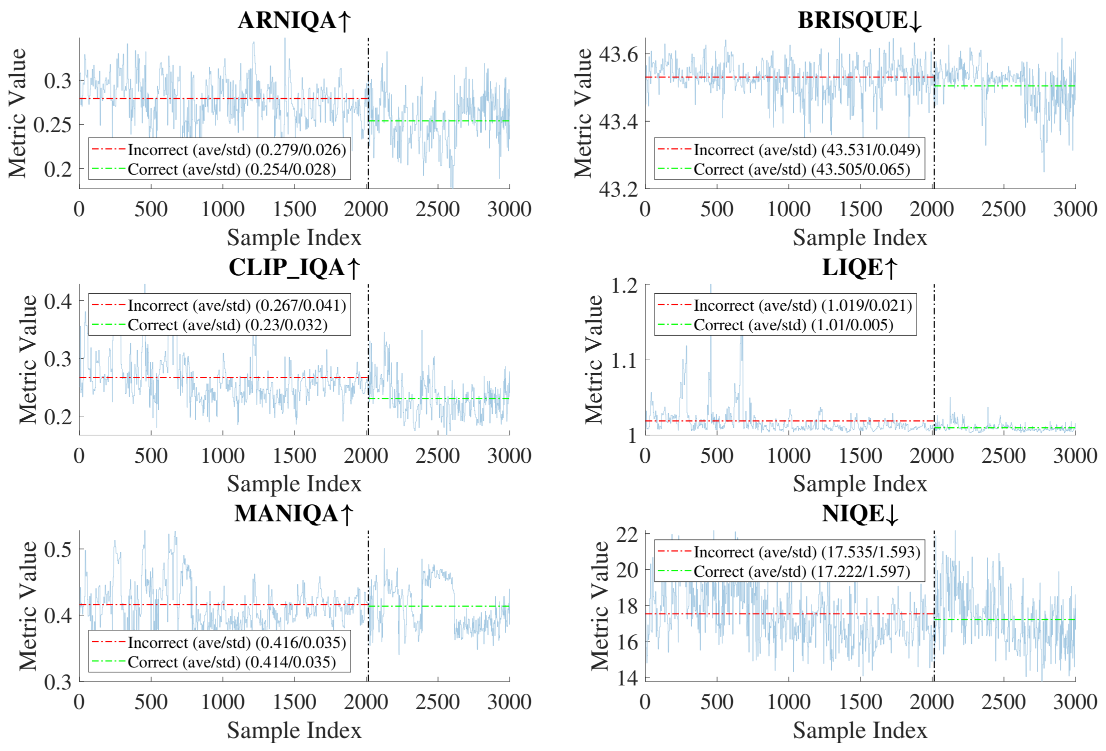


Figure C.11: NR metric scores on the Turbulence Text data processed by E$\overline{\text{D}}$VR$_s$ compared with the resulting DAN classification accuracy (32.87%)

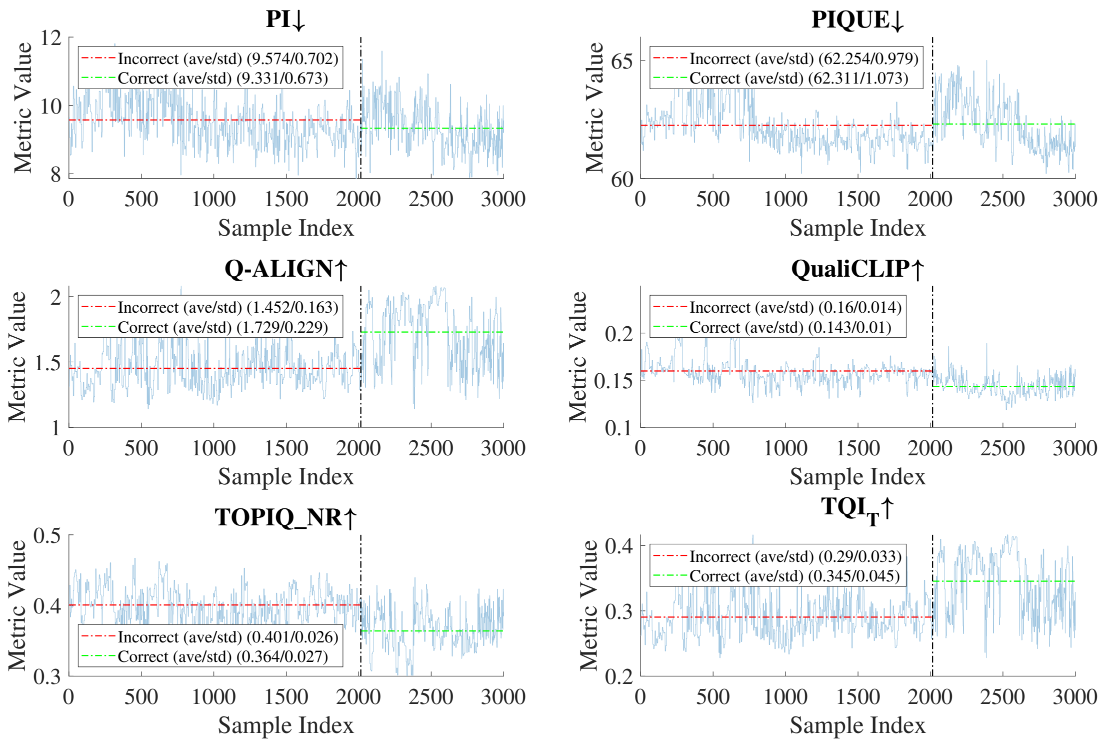


Figure C.12: NR metric scores on the Turbulence Text data processed by $\mathrm{E\overline{D}VR}_s$ compared with the resulting DAN classification accuracy (32.87%)

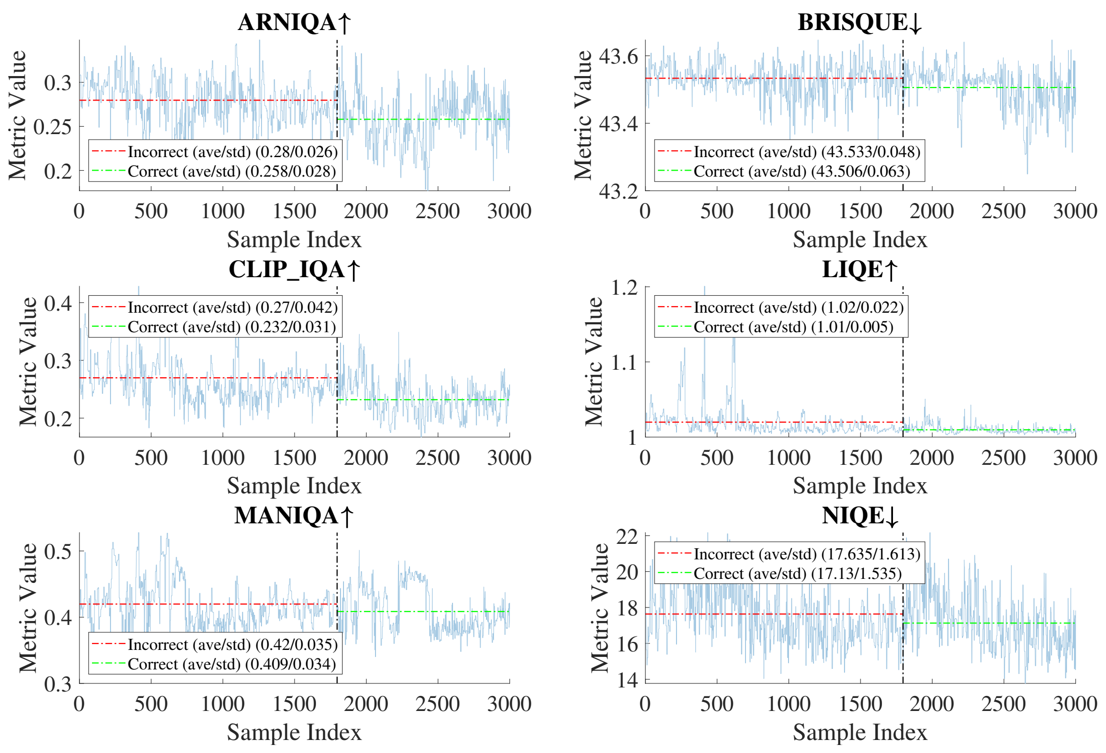


Figure C.13: NR metric scores on the Turbulence Text data processed by $\mathrm{E\overline{D}VR}_s$ compared with the resulting ASTER classification accuracy (40.13%)

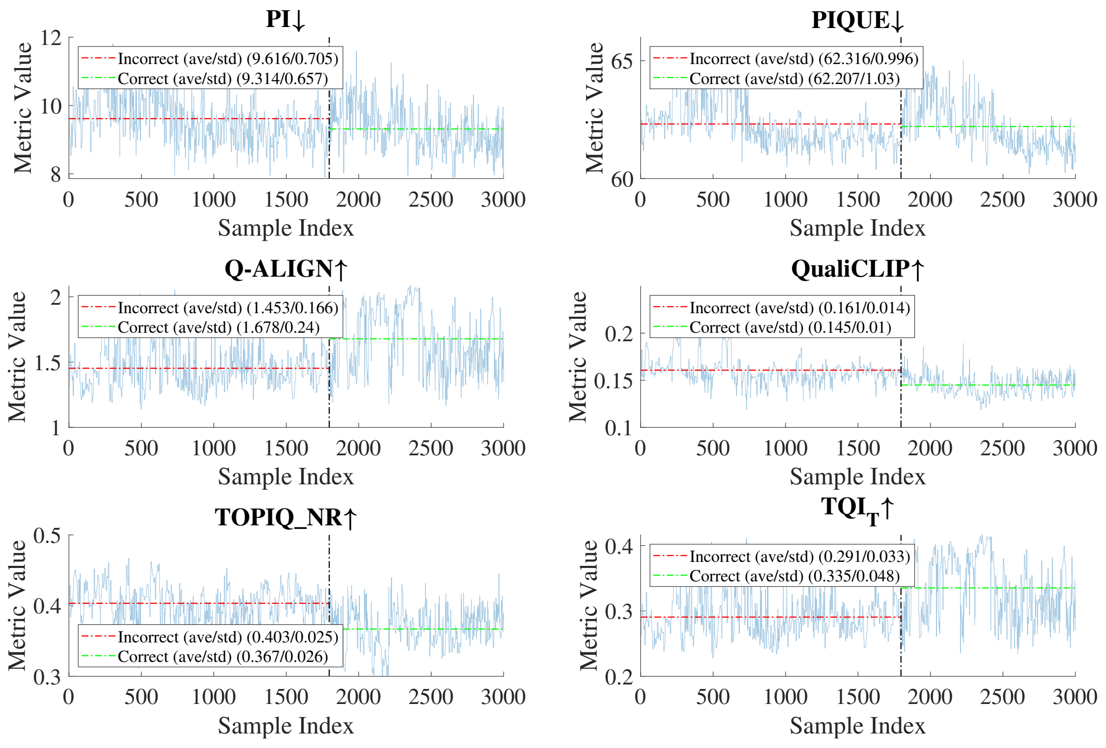


Figure C.14: NR metric scores on the Turbulence Text data processed by $\overline{\text{ED}}\text{VR}_s$ compared with the resulting ASTER classification accuracy (40.13%)

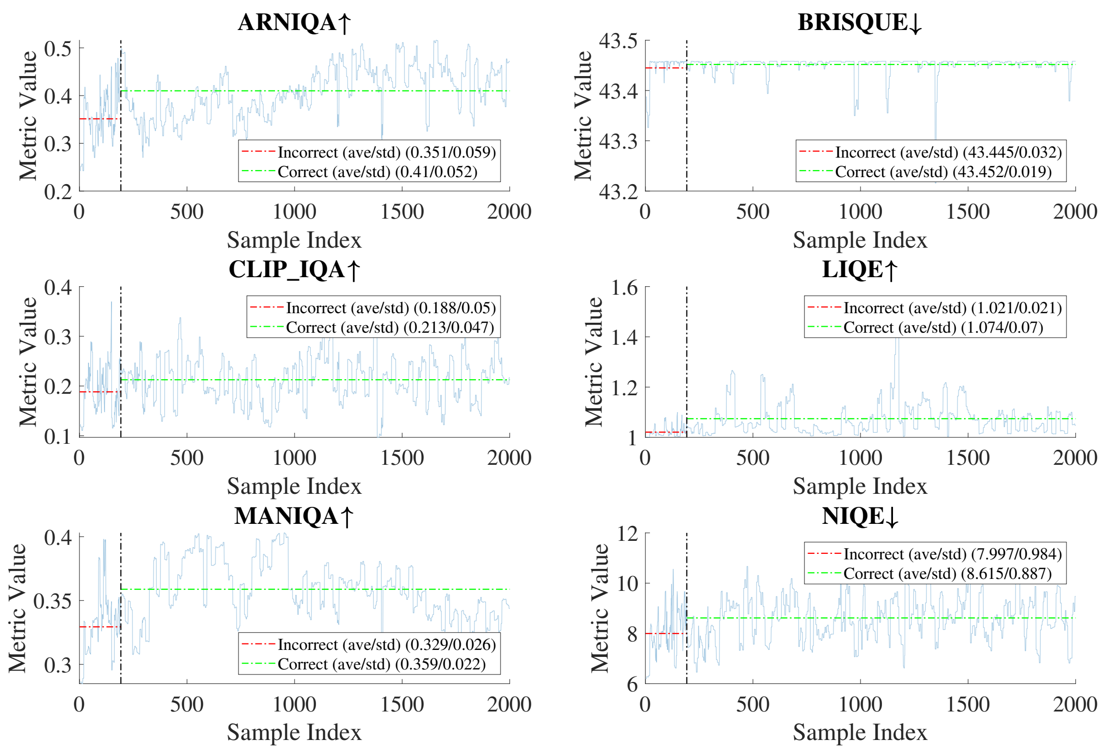


Figure C.15: NR metric scores on the Turbulence Text data processed by DATUM compared with the resulting CRNN classification accuracy (90.4%)

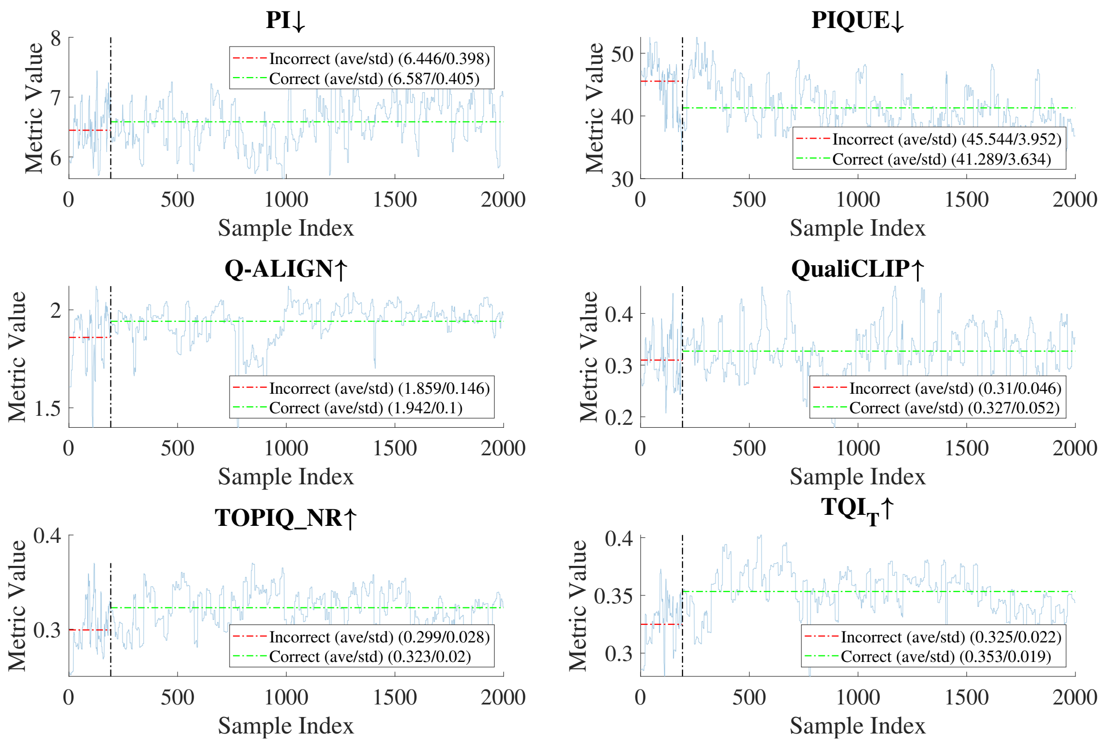


Figure C.16: NR metric scores on the Turbulence Text data processed by DATUM compared with the resulting CRNN classification accuracy (90.4%)

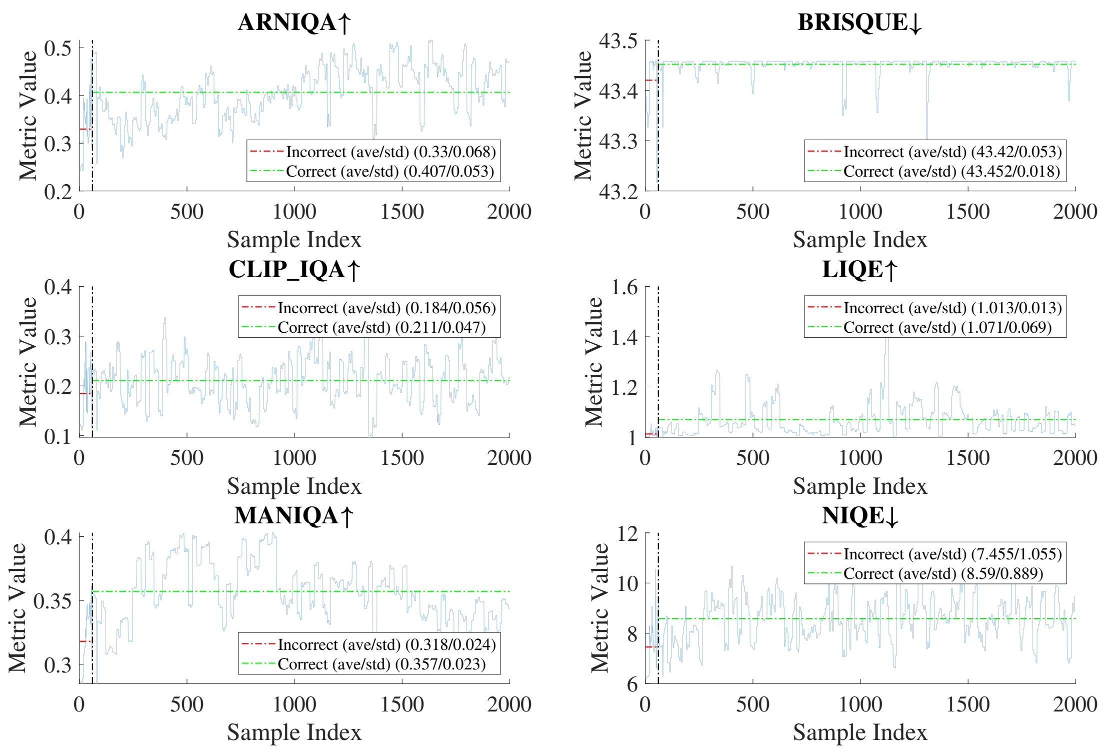


Figure C.17: NR metric scores on the Turbulence Text data processed by DATUM compared with the resulting DAN classification accuracy (97%)

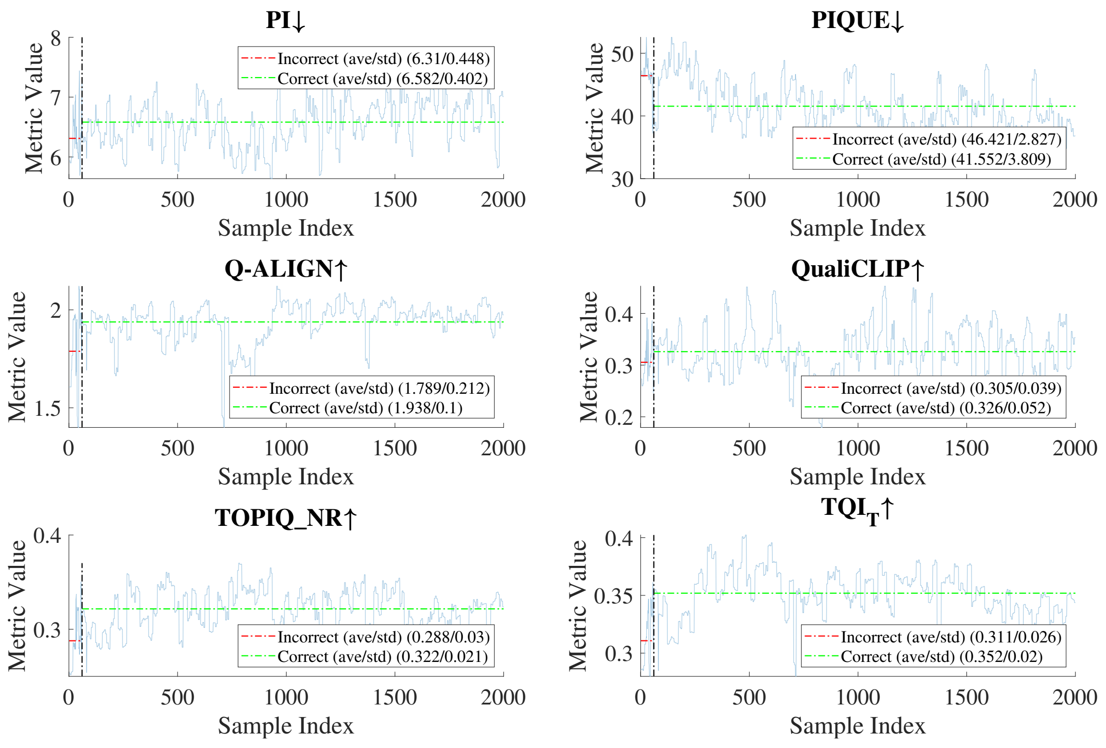


Figure C.18: NR metric scores on the Turbulence Text data processed by DATUM compared with the resulting DAN classification accuracy (97%)

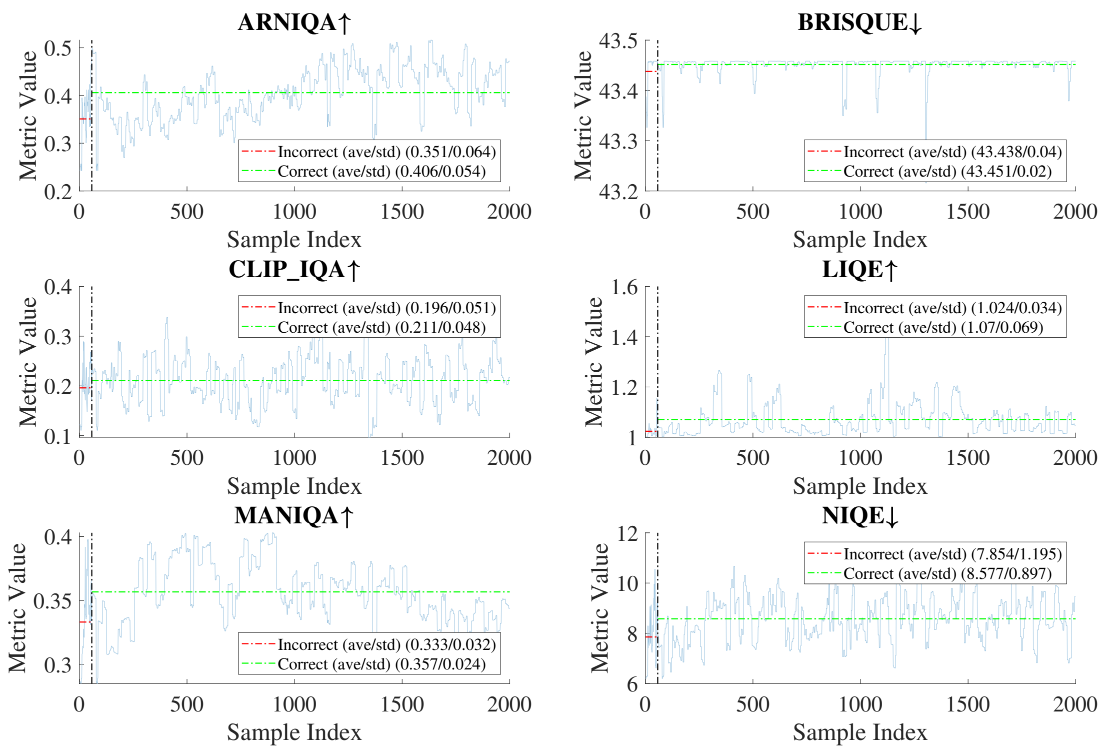


Figure C.19: NR metric scores on the Turbulence Text data processed by DATUM compared with the resulting ASTER classification accuracy (97.15%)

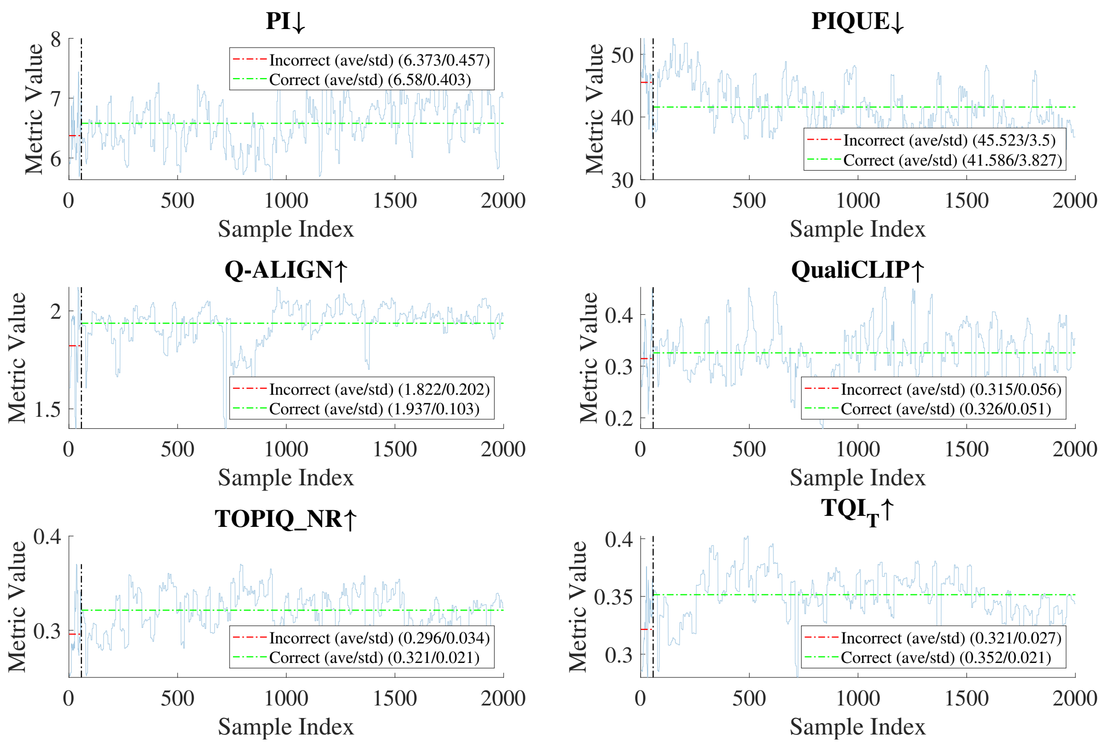


Figure C.20: NR metric scores on the Turbulence Text data processed by DATUM compared with the resulting ASTER classification accuracy (97.15%)

List of publications

Published

1. **D. Vint**, G. Di Caterina, J. Soraghan, R. Lamb, and D. Humphreys, "Evaluation of performance of vdsr super resolution on real and synthetic images," in 2019 Sensor Signal Processing for Defence Conference (SSPD), pp. 1–5, IEEE, 2019.

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

3. **D. Vint**, G. Di Caterina, P. Kirkland, R. Lamb, and D. Humphreys, "Simulation of anisoplanatic turbulence for images and videos," in 2023 Sensor Signal Processing for Defence Conference (SSPD), pp. 1–5, IEEE, 2023.

4. **D. Vint**, G. Di Caterina, P. Kirkland, and R. A. Lamb, "Deep learning-based turbulence mitigation for long range imaging," in Artificial Intelligence for Security and Defence Applications II, vol. 13206, pp. 362–377, SPIE, 2024.

5. **D. Vint**, G. Di Caterina, and R. A. Lamb, "Improving classification accuracy in post-turbulence-mitigation tasks via perceptual loss optimization," in Artificial Intelligence for Security and Defence Applications III, vol. 13679, pp. 393–411, SPIE, 2025.

References

[1] J. Barela, K. Firmanty, and M. Kastek, “Measurement and analysis of the parameters of modern long-range thermal imaging cameras,” *Sensors*, vol. 21, no. 17, p. 5700, 2021. 1

[2] X. Yang, W. Wu, K. Liu, P. W. Kim, A. K. Sangaiah, and G. Jeon, “Long-distance object recognition with image super resolution: A comparative study,” *IEEE Access*, vol. 6, pp. 13429–13438, 2018. 1

[3] R. P. Sishodia, R. L. Ray, and S. K. Singh, “Applications of remote sensing in precision agriculture: A review,” *Remote sensing*, vol. 12, no. 19, p. 3136, 2020. 1

[4] Z. Gao, C. Shen, and C. Xie, “Stacked convolutional auto-encoders for single space target image blind deconvolution,” *Neurocomputing*, vol. 313, pp. 295–305, 2018. 1, 7, 54, 55

[5] M. C. Roggemann, B. M. Welsh, and B. R. Hunt, *Imaging through turbulence.* CRC press, 1996. 1, 2, 7, 17, 75, 90

[6] T. J. Holmes, “Blind deconvolution of quantum-limited incoherent imagery: maximum-likelihood approach,” *Journal of the Optical Society of America A*, vol. 9, no. 7, pp. 1052–1061, 1992. 1

[7] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, “Deep learning for single image super-resolution: A brief review,” *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3106–3121, 2019. 1, 2, 36, 111

[8] A. Tunick, N. Tikhonov, M. Vorontsov, and G. Carhart, “Characterization of optical turbulence (cn2) data measured at the arl a_lot facility,” *US Army Research Laboratory, Adelphy, Md*, 2005. 1, 7, 27

[9] R. K. Tyson and B. W. Frazier, *Principles of adaptive optics.* CRC press, 2022. 2, 7

[10] J. E. Pearson, “Atmospheric turbulence compensation using coherent optical adaptive techniques,” *Applied optics*, vol. 15, no. 3, pp. 622–631, 1976. 2, 7

[11] N. G. Nair, K. Mei, and V. M. Patel, “At-ddpm: Restoring faces degraded by atmospheric turbulence using denoising diffusion probabilistic models,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3434–3443, 2023. 2, 7, 8, 56

[12] N. Anantrasirichai, A. Achim, and D. Bull, “Mitigating the effects of atmospheric distortion on video imagery: A review,” 2011. 2, 7, 32, 33, 74, 117

[13] N. Anantrasirichai, A. Achim, and D. Bull, “Atmospheric turbulence mitigation for sequences with moving objects using recursive image fusion,” in *2018 25th IEEE international conference on image processing (ICIP)*, pp. 2895–2899, IEEE, 2018. 26, 34

[14] C. P. Lau, Y. H. Lai, and L. M. Lui, “Variational models for joint subsampling and reconstruction of turbulence-degraded images,” *Journal of Scientific Computing*, vol. 78, pp. 1488–1525, 2019. 28, 54

[15] R. C. Hardie, M. Rucci, B. K. Karch, A. J. Dapore, D. R. Droege, and J. C. French, "Fusion of interpolated frames superresolution in the presence of atmospheric optical turbulence," *Optical Engineering*, vol. 58, no. 8, pp. 083103–083103, 2019. 34

[16] N. Chimitt, Z. Mao, G. Hong, and S. H. Chan, "Rethinking atmospheric turbulence mitigation," *arXiv preprint arXiv:1905.07498*, 2019. 28, 31, 34, 75

[17] R. C. Hardie, M. A. Rucci, B. K. Karch, A. J. Dapore, and D. R. Droege, "Super-resolution in the presence of atmospheric optical turbulence," in *Long-Range Imaging III*, vol. 10650, pp. 123–136, SPIE, 2018. 34

[18] M. A. Rucci, R. C. Hardie, and A. J. Dapore, "Comparing multiple turbulence restoration algorithms performance on noisy anisoplanatic imagery," in *Long-Range Imaging II*, vol. 10204, pp. 34–39, SPIE, 2017. 2

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012. 2, 36, 40, 50, 65, 70

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015. 2

[21] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*, pp. 10096–10106, PMLR, 2021. 2, 36, 50, 108, 111, 125, 178

[22] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3523–3542, 2021. 36, 111

[23] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5728–5739, 2022. 2, 36, 52, 56

[24] P. Piscaer, L. Knobel, L. Nijskens, M. van Lier, J. Dijk, and N. Boehrer, "Improving automatic text recognition through atmospheric turbulence," in *Artificial Intelligence for Security and Defence Applications II*, vol. 13206, pp. 378–392, SPIE, 2024. 2, 7, 26, 53

[25] W. H. Chak, C. P. Lau, and L. M. Lui, "Subsampled turbulence removal network," *arXiv preprint arXiv:1807.04418*, 2018. 3, 28, 29, 54, 55, 111, 112

[26] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016. 3, 44

[27] J. Gu, H. Cai, C. Dong, J. S. Ren, R. Timofte, Y. Gong, S. Lao, S. Shi, J. Wang, S. Yang, *et al.*, "Ntire 2022 challenge on perceptual image quality assessment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 951–967, 2022. 3, 56, 59, 60, 67, 69, 162, 172

[28] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," *Advances in neural information processing systems*, vol. 27, 2014. 4, 104, 130, 166

[29] D. Vint, G. Di Caterina, P. Kirkland, R. Lamb, and D. Humphreys, "Simulation of anisoplanatic turbulence for images and videos," in *2023 Sensor Signal Processing for Defence Conference (SSPD)*, pp. 1–5, IEEE, 2023. 4, 12, 107

[30] D. Vint, G. Di Caterina, J. Soraghan, R. Lamb, and D. Humphreys, "Evaluation of performance of vdsr super resolution on real and synthetic images," in *2019 Sensor Signal Processing for Defence Conference (SSPD)*, pp. 1–5, IEEE, 2019. 7, 27

[31] X. Zhu and J. M. Kahn, "Free-space optical communication through atmospheric turbulence channels," *IEEE Transactions on communications*, vol. 50, no. 8, pp. 1293–1300, 2002. 7

[32] A. B. Siddik, S. Sandoval, D. Voelz, L. E. Boucheron, and L. Varela, "Facial classification from multispectral imagery through atmospheric turbulence using machine learning," in *Unconventional Imaging, Sensing, and Adaptive Optics 2023*, vol. 12693, pp. 38–42, SPIE, 2023. 7

[33] G. Chen, Z. Gao, Q. Wang, and Q. Luo, "U-net like deep autoencoders for deblurring atmospheric turbulence," *Journal of Electronic Imaging*, vol. 28, no. 5, pp. 053024–053024, 2019. 7, 8, 12, 30, 55, 111, 112, 114, 115, 117, 222

[34] D. Hu and N. Anantrasirichai, "Object recognition in atmospheric turbulence scenes," in *2023 31st European Signal Processing Conference (EUSIPCO)*, pp. 561–565, IEEE, 2023. 8

[35] J. Yin, T. Sun, X. Zhang, G. Zhang, X. Wan, and J. He, "Tmts: A physics-based turbulence mitigation network guided by turbulence signature for satellite video," 2025. 7, 56

[36] G. Chen, Z. Gao, Q. Wang, and Q. Luo, "Blind de-convolution of images degraded by atmospheric turbulence," *Applied Soft Computing*, vol. 89, p. 106131, 2020. 7, 55

[37] S. Bialek, E. Bertin, S. Fabbro, H. Bouy, J.-P. Rivet, O. Lai, and J.-C. Cuillandre, "Dancecam: atmospheric turbulence mitigation in wide-field astronomical images with short-exposure video streams," *Monthly Notices of the Royal Astronomical Society*, vol. 531, no. 1, pp. 403–421, 2024. 7, 55

[38] C. P. Lau, H. Souri, and R. Chellappa, "Atfacegan: Single face image restoration and recognition from atmospheric turbulence," in *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pp. 32–39, IEEE, 2020. 7, 55

[39] K. Mei and V. M. Patel, "LTT-GAN: looking through turbulence by inverting gans," *CoRR*, vol. abs/2112.02379, 2021. 55

[40] R. Yasarla and V. M. Patel, "Cnn-based restoration of a single face image degraded by atmospheric turbulence," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 4, no. 2, pp. 222–233, 2022. 7, 55

[41] J. D. Schmidt, *Numerical Simulation of Optical Wave Propagation with examples in MATLAB*. SPIE, 2010. 8, 14, 16, 17, 18, 20, 21, 22, 24, 31, 75, 84, 98

[42] L. C. Andrews, *Laser Beam Propagation Through Random Media*. SPIE, 2005. 12, 13, 14, 15

[43] A. McFadden, "Creating a simulator of visual turbulence in long distance imaging," Master's thesis, University of Strathclyde, 2019. 8, 13

[44] O. Reynolds, "Xxix. an experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels," *Philosophical Transactions of the Royal society of London*, no. 174, pp. 935–982, 1883. 8

[45] A. N. Kolmogorov, "The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 434, no. 1890, pp. 9–13, 1991. 8, 12, 15

[46] R. C. Hardie, M. A. Rucci, A. J. Dapore, and B. K. Karch, "Block matching and wiener filtering approach to optical turbulence mitigation and its application to simulated and real imagery with quantitative error analysis," *Optical Engineering*, vol. 56, no. 7, pp. 071503–071503, 2017. 10, 30, 33, 34, 54

[47] G. B. Airy, "On the diffraction of an object-glass with circular aperture," *Transactions of the Cambridge Philosophical Society*, vol. 5, p. 283, 1835. 11

[48] M. Van Iersel and A. M. Van Eijk, "Estimating turbulence in images," in *Free-Space Laser Communications X*, vol. 7814, pp. 197–206, SPIE, 2010. 12, 27

[49] N. Chimitt and S. H. Chan, "Simulating anisoplanatic turbulence by sampling intermodal and spatially correlated zernike coefficients," *Optical Engineering*, vol. 59, no. 8, pp. 083101–083101, 2020. 16, 30, 32

[50] L. A. Bolbasova and V. Lukin, "Atmospheric research for adaptive optics," *Atmospheric and Oceanic Optics*, vol. 35, no. 3, pp. 288–302, 2022. 16

[51] N. Boehrer, P. Piscaer, G. R. Brouwer, R. Nieuwenhuizen, and J. Dijk, "Turbulence mitigation pipelines evaluation," in *Environmental Effects on Light Propagation and Adaptive Systems VI*, vol. 12731, pp. 214–230, SPIE, 2023. 16, 27, 32, 105

[52] R. C. Hardie, M. A. Rucci, S. Bose-Pillai, R. Van Hook, and B. K. Karch, "Modeling and simulation of multispectral imaging through anisoplanatic atmospheric optical turbulence," *Optical Engineering*, vol. 61, no. 9, pp. 093102–093102, 2022. 17

[53] C. Huygens, *Traite de la Lumiere.* chez Pierre Vander Aa marchand libraire, 1690. 18

[54] J. W. Goodman, *Introduction to Fourier optics.* Roberts and Company publishers, 2005. 18, 19, 23, 24, 97

[55] M. Nazarathy and J. Shamir, "First-order optics—a canonical operator representation: lossless systems," *Journal of the Optical Society of America*, vol. 72, no. 3, pp. 356–364, 1982. 20

[56] R. Nieuwenhuizen and K. Schutte, "Deep learning for software-based turbulence mitigation in long-range imaging," in *Artificial Intelligence and Machine Learning in Defense Applications*, vol. 11169, pp. 153–162, SPIE, 2019. 25, 26, 29, 54, 114, 116

[57] N. Anantrasirichai, A. Achim, N. G. Kingsbury, and D. R. Bull, "Atmospheric turbulence mitigation using complex wavelet-based fusion," *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2398–2408, 2013. 25, 26, 28, 31, 34, 63, 74, 164

[58] J. Gilles and N. B. Ferrante, "Open turbulent image set (otis)," *Pattern Recognition Letters*, vol. 86, pp. 38–41, 2017. 25, 166

[59] D. Jin, Y. Chen, Y. Lu, J. Chen, P. Wang, Z. Liu, S. Guo, and X. Bai, "Neutralizing the impact of atmospheric turbulence on complex scene imaging via deep learning," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 876–884, 2021. 26, 28, 32, 54, 55, 111

[60] Z. Mao, A. Jaiswal, Z. Wang, and S. H. Chan, "Single frame atmospheric turbulence mitigation: A benchmark study and a new physics-inspired transformer model," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIX*, pp. 430–446, Springer, 2022. 26, 28, 54, 56, 62, 112, 130, 148, 173, 178, 215

[61] S. Xu, R. Sun, Y. Chang, S. Cao, X. Xiao, and L. Yan, “Long-range turbulence mitigation: A large-scale dataset and a coarse-to-fine framework,” *arXiv preprint arXiv:2407.08377*, 2024. 26, 62

[62] R. K. Saha, E. Salcin, J. Kim, J. Smith, and S. Jayasuriya, “Turbulence strength cn2 estimation from video using physics-based deep learning,” *Optics Express*, vol. 30, no. 22, pp. 40854–40870, 2022. 27, 54, 64

[63] K. J. Miller, B. Preece, T. W. Du Bosq, and K. R. Leonard, “A data-constrained algorithm for the emulation of long-range turbulence-degraded video,” in *Infrared imaging systems: design, analysis, modeling, and testing XXX*, vol. 11001, pp. 204–214, SPIE, 2019. 27, 29

[64] M. Hirsch, S. Sra, B. Schölkopf, and S. Harmeling, “Efficient filter flow for space-variant multiframe blind deconvolution,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 607–614, IEEE, 2010. 28, 31

[65] C. Zhang, B. Xue, F. Zhou, and W. Xiong, “Removing atmospheric turbulence effects in unified complex steerable pyramid framework,” *IEEE Access*, vol. 6, pp. 75855–75867, 2018. 28, 29

[66] J. Gao, N. Anantrasirichai, and D. Bull, “Atmospheric turbulence removal using convolutional neural network,” *arXiv preprint arXiv:1912.11350*, 2019. 28, 30, 54, 55, 111, 112, 114, 116

[67] H. van der Elst and J. van Schalkwyk, “Modelling and restoring images distorted by atmospheric turbulence,” in *Proceedings of COMSIG’94-1994 South African Symposium on Communications and Signal Processing*, pp. 162–167, IEEE, 1994. 28

[68] N. Boehrer, R. P. Nieuwenhuizen, and J. Dijk, “Turbulence mitigation in imagery including moving objects from a static event camera,” *Optical Engineering*, vol. 60, no. 5, pp. 053101–053101, 2021. 28

[69] S. H. Chan, “Tilt-then-blur or blur-then-tilt? clarifying the atmospheric turbulence model,” *IEEE Signal Processing Letters*, vol. 29, pp. 1833–1837, 2022. 29, 181

[70] X. Zhu and P. Milanfar, “Removing atmospheric turbulence via space-invariant deconvolution,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 157–170, 2012. 29, 34

[71] E. Repasi and R. Weiss, “Analysis of image distortions by atmospheric turbulence and computer simulation of turbulence effects,” in *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XIX*, vol. 6941, pp. 256–268, SPIE, 2008. 29

[72] E. Repasi and R. Weiss, “Computer simulation of image degradations by atmospheric turbulence for horizontal views,” in *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXII*, vol. 8014, pp. 279–287, SPIE, 2011. 29, 75

[73] K. R. Leonard, J. Howe, and D. E. Oxford, “Simulation of atmospheric turbulence effects and mitigation algorithms on stand-off automatic facial recognition,” in *Optics and Photonics for Counterterrorism, Crime Fighting, and Defence VIII*, vol. 8546, pp. 182–198, SPIE, 2012. 29

[74] M. A. Rucci, R. C. Hardie, R. K. Martin, and S. Gladysz, “Atmospheric optical turbulence mitigation using iterative image registration and least squares lucky look fusion,” *Applied Optics*, vol. 61, no. 28, pp. 8233–8247, 2022. 30, 33

[75] J. Cheng, J. Li, C. Dai, Y. Ren, G. Xu, S. Li, X. Chen, and W. Zhu, "Research on atmospheric turbulence-degraded image restoration based on generative adversarial networks," in *First International Conference on Spatial Atmospheric Marine Environmental Optics (SAME 2023)*, vol. 12706, pp. 37–44, SPIE, 2023. 30, 55, 162

[76] Z. Mao, N. Chimitt, and S. H. Chan, "Accelerating atmospheric turbulence simulation via learned phase-to-space transform," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14759–14768, 2021. 30, 32, 55, 62, 75, 76, 93, 102, 103, 104, 105, 227, 228, 229, 230

[77] N. Chimitt, X. Zhang, Z. Mao, and S. H. Chan, "Real-time dense field phase-to-space simulation of imaging through atmospheric turbulence," *IEEE Transactions on Computational Imaging*, vol. 8, pp. 1159–1169, 2022. 30, 75, 127

[78] N. Chimitt and S. H. Chan, "Anisoplanatic optical turbulence simulation for near-continuous cn2 profiles without wave propagation," *arXiv preprint arXiv:2305.09036*, 2023. 30

[79] J. P. Bos and M. C. Roggemann, "Technique for simulating anisoplanatic image formation over long horizontal paths," *Optical Engineering*, vol. 51, no. 10, pp. 101704–101704, 2012. 30, 75

[80] J. Beck, C. Bekins, and J. P. Bos, "Wavepy: a python package for wave optics," in *Long-Range Imaging*, vol. 9846, pp. 8–17, SPIE, 2016. 31

[81] R. C. Hardie, J. D. Power, D. A. LeMaster, D. R. Droege, S. Gladysz, and S. Bose-Pillai, "Simulation of anisoplanatic imaging through optical turbulence using numerical wave propagation with new validation analysis," *Optical Engineering*, vol. 56, no. 7, p. 071502, 2017. 31, 75, 77, 82, 87, 89, 92, 106, 107

[82] N. Anugu and J. Lancelot, "Study of atmospheric turbulence with shack hartmann wavefront sensor," *Journal of Optics*, vol. 42, pp. 128–140, 2013.

[83] D. Wegner, "Turbulence simulation for anisoplanatic imaging based on phase screens with experimental validation of differential tilt variances," in *Electro-Optical and Infrared Systems: Technology and Applications XVIII and Electro-Optical Remote Sensing XV*, vol. 11866, pp. 108–127, SPIE, 2021. 31

[84] S. N. Rai and C. Jawahar, "Learning to generate atmospheric turbulent images," in *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics*, pp. 261–271, Springer, 2019. 31, 75

[85] A. Schwartzman, M. Alterman, R. Zamir, and Y. Y. Schechner, "Turbulence-induced 2d correlated image distortion," in *2017 IEEE International Conference on Computational Photography (ICCP)*, pp. 1–13, IEEE, 2017. 31, 75

[86] G. Franz, D. Wegner, J. Pérez, and S. Kessler, "Towards atmospheric turbulence simulation using a conditional variational autoencoder," in *Electro-Optical and Infrared Systems: Technology and Applications XVIII and Electro-Optical Remote Sensing XV*, vol. 11866, pp. 108–121, SPIE, 2021. 31

[87] X. Zhang, Z. Mao, N. Chimitt, and S. H. Chan, "Imaging through the atmosphere using turbulence mitigation transformer," *IEEE Transactions on Computational Imaging*, 2024. 32, 54, 56, 62, 111

[88] X. Zhang, N. Chimitt, Y. Chi, Z. Mao, and S. H. Chan, "Spatio-temporal turbulence mitigation: A translational perspective," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2889–2899, 2024. 32, 54, 56, 62, 63, 112, 113, 125, 126, 130, 147, 158, 164, 173, 174, 178, 187

[89] R. Gregory, "A technique for minimizing the effects of atmospheric disturbance on photographic telescopes," *Nature*, vol. 203, no. 4942, pp. 274–275, 1964. 32

[90] C. S. Huebner and C. Scheifling, "Software-based mitigation of image degradation due to atmospheric turbulence," in *Optics in Atmospheric Propagation and Adaptive Systems XIII*, vol. 7828, pp. 198–209, SPIE, 2010. 33, 34

[91] N. Joshi and M. F. Cohen, "Seeing mt. rainier: Lucky imaging for multi-image denoising, sharpening, and haze removal," in *2010 IEEE International Conference on Computational Photography (ICCP)*, pp. 1–8, IEEE, 2010. 33

[92] Z. Wen, D. Fraser, and A. Lambert, "Bicoherence used to predict lucky regions in turbulence affected surveillance," in *2006 IEEE International Conference on Video and Signal Based Surveillance*, pp. 108–108, IEEE, 2006. 33, 34

[93] Z. Mao, N. Chimitt, and S. H. Chan, "Image reconstruction of static and dynamic scenes through anisoplanatic turbulence," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 1415–1428, 2020. 33, 34

[94] R. He, Z. Wang, Y. Fan, and D. Fengg, "Atmospheric turbulence mitigation based on turbulence extraction," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1442–1446, IEEE, 2016. 33

[95] C. P. Lau, Y. H. Lai, and L. M. Lui, "Restoration of atmospheric turbulence-distorted images via rpca and quasiconformal maps," *Inverse Problems*, vol. 35, no. 7, p. 074002, 2019. 33

[96] C. S. Huebner, "Compensating image degradation due to atmospheric turbulence in anisoplanatic conditions," in *Mobile Multimedia/Image Processing, Security, and Applications 2009*, vol. 7351, pp. 43–53, SPIE, 2009. 33

[97] X. Zhu and P. Milanfar, "Image reconstruction from videos distorted by atmospheric turbulence," in *Visual Information Processing and Communication*, vol. 7543, pp. 228–235, SPIE, 2010. 34

[98] C. Bondeau and E. Bourennane, "Restoration of images degraded by the atmospheric turbulence," in *ICSP'98. 1998 Fourth International Conference on Signal Processing (Cat. No. 98TH8344)*, vol. 2, pp. 1056–1059, IEEE, 1998. 34

[99] R. C. Hardie and D. A. LeMaster, "On the simulation and mitigation of anisoplanatic optical turbulence for long range imaging," in *Long-Range Imaging II*, vol. 10204, pp. 40–59, SPIE, 2017. 34

[100] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943. 39

[101] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958. 39

[102] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986. 39

[103] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. 39, 51

[104] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014. 39

[105] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 39

[106] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 2002. 40, 41, 42

[107] R. S. Thakur, R. N. Yadav, and L. Gupta, “State-of-art analysis of image denoising methods using convolutional neural networks,” *IET Image Processing*, vol. 13, no. 13, pp. 2367–2380, 2019. 42, 116

[108] C. Dong, C. C. Loy, and X. Tang, “Accelerating the super-resolution convolutional neural network,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 391–407, Springer, 2016. 42

[109] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, “Ntire 2017 challenge on single image super-resolution: Methods and results,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 114–125, 2017. 42

[110] J. Cai, S. Gu, R. Timofte, and L. Zhang, “Ntire 2019 challenge on real image super-resolution: Methods and results,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019. 42, 111, 112, 115

[111] R. Jin and Q. Niu, “Automatic fabric defect detection based on an improved yolov5,” *Mathematical Problems in Engineering*, vol. 2021, no. 1, p. 7321394, 2021. 43

[112] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Deep laplacian pyramid networks for fast and accurate super-resolution,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 624–632, 2017. 44

[113] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014. 46

[114] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 48, 50, 65, 66, 69

[115] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network [c],” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 2017, pp. 4681–4690, 2017. 48, 59

[116] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017. 48

[117] Z. Wang and S. Ji, “Smoothed dilated convolutions for improved dense prediction,” in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2486–2495, 2018. 49

[118] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017. 49

[119] X. Zhu, H. Hu, S. Lin, and J. Dai, “Deformable convnets v2: More deformable, better results,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9308–9316, 2019. 49, 119, 121

[120] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 50, 56, 65, 70, 122, 129

[121] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015. 50, 51, 54, 111

[122] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014. 51

[123] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014. 51

[124] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017. 51

[125] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. 51, 65, 67

[126] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021. 52

[127] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PmLR, 2021. 52, 53, 65, 66, 67

[128] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020. 53

[129] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International conference on machine learning*, pp. 8821–8831, Pmlr, 2021. 53

[130] K. Kee, C. Wu, D. A. Paulson, and C. C. Davis, "Assisting target recognition through strong turbulence with the help of neural networks," *Applied Optics*, vol. 59, no. 30, pp. 9434–9442, 2020. 53

[131] D. Hu and N. Anantrasirichai, "Object recognition in atmospheric turbulence scenes," *arXiv preprint arXiv:2210.14318*, 2022. 53

[132] D. A. LeMaster, S. Leung, and O. L. Mendoza-Schrock, "Joint object classification and turbulence strength estimation using convolutional neural networks," *Applied Optics*, vol. 60, no. 25, pp. G40–G48, 2021. 53

[133] U. Celik, H. A. Yasar, M. Yigitoglu Keskin, C. Bayar, I. Aslantas, and Y. Midilli, "Estimation of ground-based atmospheric turbulence strength (c n 2) by neural network architecture," *Applied Optics*, vol. 63, no. 28, pp. 7402–7409, 2024. 54

[134] M. A. Hoffmire, R. C. Hardie, M. A. Rucci, R. Van Hook, and B. K. Karch, "Deep learning for anisoplanatic optical turbulence mitigation in long-range imaging," *Optical Engineering*, vol. 60, no. 3, pp. 033103–033103, 2021. 54

[135] A. W. van Eekeren, J. Dijk, and K. Schutte, "Turbulence mitigation methods and their evaluation," in *Electro-Optical and Infrared Systems: Technology and Applications XI*, vol. 9249, pp. 206–212, SPIE, 2014. 54

[136] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017. 54, 114, 115, 117, 223

[137] H. Fazlali, S. Shirani, M. BradforSd, and T. Kirubarajan, "Atmospheric turbulence removal in long-range imaging using a data-driven-based approach," *International Journal of Computer Vision*, vol. 130, no. 4, pp. 1031–1049, 2022. 54

[138] N. G. Nair and V. M. Patel, "Confidence guided network for atmospheric turbulence mitigation," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1359–1363, IEEE, 2021. 54, 55

[139] N. Anantrasirichai, "Atmospheric turbulence removal with complex-valued convolutional neural network," *Pattern Recognition Letters*, vol. 171, pp. 69–75, 2023. 54, 55

[140] S. Nah, R. Timofte, S. Baik, S. Hong, G. Moon, S. Son, and K. Mu Lee, "Ntire 2019 challenge on video deblurring: Methods and results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019. 54, 117

[141] Y. Lou, S. H. Kang, S. Soatto, and A. L. Bertozzi, "Video stabilization of atmospheric turbulence distortion," *Inverse Problems and Imaging*, vol. 7, no. 3, pp. 839–861, 2013. 54

[142] T. Jain, M. Lubien, and J. Gilles, "Evaluation of neural network algorithms for atmospheric turbulence mitigation," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXXI*, vol. 12122, pp. 223–236, SPIE, 2022. 54, 111

[143] C. Su, X. Wu, and Y. Guo, "Restoration of turbulence-degraded images using the modified convolutional neural network," *Applied Intelligence*, vol. 53, no. 5, pp. 5834–5844, 2023. 55, 162

[144] X. Liu, G. Li, Z. Zhao, Q. Cao, Z. Zhang, S. Yan, J. Xie, and M. Tang, "Eaf-wgan: Enhanced alignment fusion-wasserstein generative adversarial network for turbulent image restoration," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023. 54, 55, 62, 111

[145] S. N. Rai and C. Jawahar, "Removing atmospheric turbulence via deep adversarial learning," *IEEE Transactions on Image Processing*, vol. 31, pp. 2633–2646, 2022. 54, 55, 63, 164

[146] Z. Liu, P. Hill, and N. Anantrasirichai, "Jdatt: A joint distillation framework for atmospheric turbulence mitigation and target detection," *arXiv preprint arXiv:2507.19780*, 2025.

[147] X. Zhang, N. Chimitt, X. Wang, Y. Yuan, and S. H. Chan, "Learning phase distortion with selective state space models for video turbulence mitigation," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 2127–2138, 2025. 56

[148] Y. Xia, C. Zhou, C. Zhu, C. Xu, and B. Shi, "Planet: Learning to mitigate atmospheric turbulence in planetary images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 8584–8592, 2025. 55

[149] P. Hill, Z. Liu, and N. Anantrasirichai, "Mamat: 3d mamba-based atmospheric turbulence removal and its object detection capability," *arXiv preprint arXiv:2503.17700*, 2025. 54, 56

[150] X. Li, H. Ma, and J. Chu, "The design of a method for recovering degraded images of atmospheric turbulence based on deep learning," in *International Conference on Optical Technology, Semiconductor Materials, and Devices (OTSMD 2022)*, vol. 12589, pp. 136–142, SPIE, 2023. 55, 162

[151] C. Su, X. Wu, Y. Guo, S. Zhang, Z. Wang, and D. Shi, "Atmospheric turbulence degraded image restoration using a modified dilated convolutional network," *IET Image Processing*, vol. 16, no. 13, pp. 3507–3517, 2022. 55

[152] Y. Guo, X. Wu, C. Qing, C. Su, Q. Yang, and Z. Wang, "Blind restoration of images distorted by atmospheric turbulence based on deep transfer learning," in *Photonics*, vol. 9, p. 582, MDPI, 2022. 55, 62

[153] C. Whyte, *Characterizing Atmospheric Turbulence and Removing Distortion in Long-range Imaging*. PhD thesis, ARIZONA STATE UNIVERSITY, 2021. 55

[154] S. A. Khowaja, I. H. Lee, and J. Yoon, "2nd place solutions for ug2+ challenge 2022-d3 net for mitigating atmospheric turbulence from images," *arXiv preprint arXiv:2208.12332*, 2022. 55, 62

[155] S. A. Khowaja, U. Ali, K. Dev, and I. H. Lee, "Image registration and deep neurofuzzy networks for mitigating atmospheric turbulence effects in consumer-based optical imaging," *IEEE Transactions on Consumer Electronics*, 2025. 55

[156] Y. Lu, L. Zhou, *et al.*, "Research on turbulence-removal optical imaging based on multi-scale gan and sequential images," 2025. 55

[157] B. Y. Feng, M. Xie, and C. A. Metzler, "Turbugan: An adversarial learning approach to spatially-varying multiframe blind deconvolution with applications to imaging through turbulence," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 3, pp. 543–556, 2022. 55

[158] X. Wang, S. López-Tapia, and A. K. Katsaggelos, "Atmospheric turbulence correction via variational deep diffusion," *arXiv preprint arXiv:2305.05077*, 2023. 56, 62

[159] G. Sun, Q. Ma, L. Zhang, H. Wang, Z. Gao, and H. Zhang, "Probabilistic prior driven attention mechanism based on diffusion model for imaging through atmospheric turbulence," *arXiv preprint arXiv:2411.10321*, 2024. 56

[160] D. Yang, Z. Chen, Y. Zhang, X. Wu, O. Sasaki, and J. Pu, "Restoration of atmospheric turbulence-degraded images based on multi-frame information," *Optics Express*, vol. 33, no. 1, pp. 369–383, 2025. 56

[161] M. A. Kiasari, K. Muhammad, S. Bakshi, and I. H. Lee, "Hybrid transformer-cnn-based attention in video turbulence mitigation (hatm)," in *International Conference on Pattern Recognition*, pp. 242–256, Springer, 2025. 56

[162] Z. Liu, Z. Zhao, Y. Yuan, Z. Qiao, J. Bai, and Z. Ji, "1st place solutions for ug2+ challenge 2022 atmospheric turbulence mitigation," *arXiv preprint arXiv:2210.16847*, 2022. 56, 62

[163] R. K. Saha, D. Qin, N. Li, J. Ye, and S. Jayasuriya, "Turb-seg-res: A segment-then-restore pipeline for dynamic videos with atmospheric turbulence," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25286–25296, 2024. 56, 63

[164] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023. 56

[165] G. Jinjin, C. Haoming, C. Haoyu, Y. Xiaoxing, J. S. Ren, and D. Chao, "Pipal: a large-scale image quality assessment dataset for perceptual image restoration," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 633–651, Springer, 2020. 57, 59, 60, 61, 162, 163, 165, 166, 173, 177

[166] H. Lin, V. Hosu, and D. Saupe, "Kadid-10k: A large-scale artificially distorted iqa database," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–3, IEEE, 2019. 57, 62, 173

[167] K. Pearson, "Vii. note on regression and inheritance in the case of two parents," *proceedings of the royal society of London*, vol. 58, no. 347-352, pp. 240–242, 1895. 57

[168] C. Spearman, "The proof and measurement of association between two things.," 1961. 57

[169] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1-2, pp. 81–93, 1938. 57

[170] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*, pp. 1015–1021, Springer, 2006. 58

[171] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004. 59, 65, 72, 117

[172] Y. Blau and T. Michaeli, "The perception-distortion tradeoff," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6228–6237, 2018. 59, 163, 213

[173] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *Proceedings of the IEEE international conference on computer vision*, pp. 4491–4500, 2017. 59

[174] C. Chen, J. Mo, J. Hou, H. Wu, L. Liao, W. Sun, Q. Yan, and W. Lin, "Topiq: A top-down approach from semantics to distortions for image quality assessment," *IEEE Transactions on Image Processing*, 2024. 59, 65, 69, 72

[175] B. B. Moser, A. S. Shanbhag, F. Raue, S. Frolov, S. Palacio, and A. Dengel, "Diffusion models, image super-resolution, and everything: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2024. 59

[176] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5485–5493, 2017. 59

[177] R. Dahl, M. Norouzi, and J. Shlens, "Pixel recursive super resolution," in *Proceedings of the IEEE international conference on computer vision*, pp. 5439–5448, 2017. 59

[178] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*, pp. 2366–2369, IEEE, 2010. 59

[179] J. Gu, H. Cai, C. Dong, J. S. Ren, Y. Qiao, S. Gu, and R. Timofte, "Ntire 2021 challenge on perceptual image quality assessment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 677–690, 2021. 60, 61, 172

[180] S. Yang, T. Wu, S. Shi, S. Lao, Y. Gong, M. Cao, J. Wang, and Y. Yang, "Maniqa: Multi-dimension attention network for no-reference image quality assessment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1191–1200, 2022. 60, 65, 67, 125

[181] S. Lao, Y. Gong, S. Shi, S. Yang, T. Wu, J. Wang, W. Xia, and Y. Yang, "Attentions help cnns see better: Attention-based hybrid image quality assessment network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1140–1149, 2022. 60, 65, 69

[182] X. Liu, X. Min, W. Sun, Y. Zhang, K. Zhang, R. Timofte, G. Zhai, Y. Gao, Y. Cao, T. Kou, *et al.*, "Ntire 2023 quality assessment of video enhancement challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1551–1569, 2023. 60

[183] X. Liu, X. Min, G. Zhai, C. Li, T. Kou, W. Sun, H. Wu, Y. Gao, Y. Cao, Z. Zhang, *et al.*, "Ntire 2024 quality assessment of ai-generated content challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6337–6362, 2024. 60

[184] V. Hosu, M. V. Conde, L. Agnolucci, N. Barman, S. Zadtootaghaj, and R. Timofte, "Aim 2024 challenge on uhd blind photo quality assessment," *arXiv preprint arXiv:2409.16271*, 2024. 60, 164

[185] V. Hosu, L. Agnolucci, O. Wiedemann, D. Iso, and D. Saupe, "Uhd-iqa benchmark database: Pushing the boundaries of blind photo quality assessment," *arXiv preprint arXiv:2406.17472*, 2024. 60, 69, 173, 177

[186] L. Agnolucci, L. Galteri, and M. Bertini, "Quality-aware image-text alignment for real-world image quality assessment," *arXiv preprint arXiv:2403.11176*, 2024. 60, 65, 69, 164

[187] W. Sun, W. Zhang, Y. Cao, L. Cao, J. Jia, Z. Chen, Z. Zhang, X. Min, and G. Zhai, "Assessing uhd image quality from aesthetics, distortions, and saliency," *arXiv preprint arXiv:2409.00749*, 2024. 60

[188] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, *et al.*, "Image database tid2013: Peculiarities, results and perspectives," *Signal processing: Image communication*, vol. 30, pp. 57–77, 2015. 61, 173, 177

[189] J. Gu, H. Cai, H. Chen, X. Ye, J. Ren, and C. Dong, "Image quality assessment for perceptual image restoration: A new dataset, benchmark and metric," *arXiv preprint arXiv:2011.15002*, 2020. 60, 71

[190] E. Prashnani, H. Cai, Y. Mostofi, and P. Sen, "Pieapp: Perceptual image-error assessment through pairwise preference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1808–1817, 2018. 61, 65, 71

[191] S. Bosse, D. Maniry, K.-R. Müller, T. Wiegand, and W. Samek, "Deep neural networks for no-reference and full-reference image quality assessment," *IEEE Transactions on image processing*, vol. 27, no. 1, pp. 206–219, 2017. 61

[192] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti, "Tid2008-a database for evaluation of full-reference visual quality assessment metrics," *Advances of modern radioelectronics*, vol. 10, no. 4, pp. 30–45, 2009. 61, 165

[193] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment," *IEEE transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, 2011. 61, 65, 70

[194] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on image processing*, vol. 15, no. 11, pp. 3440–3451, 2006. 62, 66

[195] D. Ghadiyaram and A. C. Bovik, “Massive online crowdsourced study of subjective and objective picture quality,” *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 372–387, 2015. 62

[196] E. C. Larson and D. M. Chandler, “Most apparent distortion: full-reference image quality assessment and the role of strategy,” *Journal of electronic imaging*, vol. 19, no. 1, pp. 011006–011006, 2010. 62, 173

[197] V. Hosu, H. Lin, T. Sziranyi, and D. Saupe, “Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment,” *IEEE Transactions on Image Processing*, vol. 29, pp. 4041–4056, 2020. 62, 173

[198] H. Wei, K. Gao, J. Wang, Q. Tang, X. Tang, and F. Xu, “Optical imaging degradation simulation and transformer-based image restoration for remote sensing,” *IEEE Geoscience and Remote Sensing Letters*, 2024. 62

[199] C. Wang, G. Sun, C. Wang, Z. Gao, and H. Wang, “Monte carlo-based restoration of images degraded by atmospheric turbulence,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.

[200] D. Lao, C. Wang, A. Wong, and S. Soatto, “Diffeomorphic template registration for atmospheric turbulence mitigation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25107–25116, 2024.

[201] Y. Xia, C. Zhou, C. Zhu, M. Teng, C. Xu, and B. Shi, “Nb-gtr: Narrow-band guided turbulence removal,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24934–24943, 2024.

[202] W. Jiang, V. Boominathan, and A. Veeraraghavan, “Nert: Implicit neural representations for general unsupervised turbulence mitigation,” *arXiv preprint arXiv:2308.00622*, 2023. 62

[203] Z. Zou and N. Anantrasirichai, “Deturb: Atmospheric turbulence mitigation with deformable 3d convolutions and 3d swin transformers,” *arXiv preprint arXiv:2407.20855*, 2024. 62

[204] X. Wang, *Deep Learning Applied to Image and Video Processing*. PhD thesis, Northwestern University, 2024. 62, 63

[205] X. Liu, L. Tang, G. Li, Z. Zhang, S. Yan, Y. Xiao, J. Xie, and M. Tang, “Idssi: Image deturbulence with semantic and spatial–temporal information,” *Pattern Recognition*, vol. 156, p. 110813, 2024. 62

[206] V. Philippe and T. Bourlai, “Exploring image augmentation methods for long-distance face recognition using deep learning,” in *SoutheastCon 2024*, pp. 1144–1150, IEEE, 2024. 63

[207] D. N. Groff, K. J. Miller, and T. W. Du Bosq, “Towards development of improved metrics for quantifying turbulence imposed degradation of long-range video,” in *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXXII*, vol. 11740, pp. 139–148, SPIE, 2021. 63

[208] E. Sharghi, K. Mcbryde, E. Walter, B. N. Campbell, and S. Hammel, “Machine-learning-based image quality prediction from optical turbulence and meteorological observations,” in *Laser Communication and Propagation through the Atmosphere and Oceans XIII*, vol. 13147, pp. 209–218, SPIE, 2024. 63

[209] P. Hill, N. Anantrasirichai, A. Achim, and D. Bull, “Atmospheric turbulence removal with video sequence deep visual priors,” *arXiv preprint arXiv:2402.19041*, 2024. 64

[210] C. Chen and J. Mo, “IQA-PyTorch: Pytorch toolbox for image quality assessment.” [Online]. Available: https://github.com/chaofengc/IQA-PyTorch, 2022. 64, 65

[211] L. Agnolucci, L. Galteri, M. Bertini, and A. Del Bimbo, “Arniqa: Learning distortion manifold for image quality assessment,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 189–198, 2024. 65, 66

[212] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PmLR, 2020. 65

[213] A. Mittal, A. K. Moorthy, and A. C. Bovik, “No-reference image quality assessment in the spatial domain,” *IEEE Transactions on image processing*, vol. 21, no. 12, pp. 4695–4708, 2012. 65, 66, 125

[214] J. Wang, K. C. Chan, and C. C. Loy, “Exploring clip for assessing the look and feel of images,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, pp. 2555–2563, 2023. 65, 66, 125

[215] W. Zhang, G. Zhai, Y. Wei, X. Yang, and K. Ma, “Blind image quality assessment via vision-language correspondence: A multitask learning perspective,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14071–14081, 2023. 65, 67

[216] A. Mittal, R. Soundararajan, and A. C. Bovik, “Making a “completely blind” image quality analyzer,” *IEEE Signal processing letters*, vol. 20, no. 3, pp. 209–212, 2012. 65, 67, 125

[217] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, “The 2018 pirm challenge on perceptual image super-resolution,” in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018. 65, 68

[218] C. Ma, C.-Y. Yang, X. Yang, and M.-H. Yang, “Learning a no-reference quality metric for single-image super-resolution,” *Computer Vision and Image Understanding*, vol. 158, pp. 1–16, 2017. 65, 68

[219] N. Venkatanath, D. Praneeth, M. C. Bh, S. S. Channappayya, and S. S. Medasani, “Blind image quality evaluation using perception based features,” in *2015 twenty first national conference on communications (NCC)*, pp. 1–6, IEEE, 2015. 65, 68

[220] H. Wu, Z. Zhang, W. Zhang, C. Chen, L. Liao, C. Li, Y. Gao, A. Wang, E. Zhang, W. Sun, *et al.*, “Q-align: Teaching lmms for visual scoring via discrete text-defined levels,” *arXiv preprint arXiv:2312.17090*, 2023. 65, 68

[221] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, “Complex wavelet structural similarity: A new image similarity index,” *IEEE transactions on image processing*, vol. 18, no. 11, pp. 2385–2401, 2009. 65, 70, 125

[222] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, “Image quality assessment: Unifying structure and texture similarity,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 5, pp. 2567–2581, 2020. 65, 70

[223] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018. 65, 70

[224] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–1402, Ieee, 2003. 65, 71

[225] Q. Ye, H. Xu, J. Ye, M. Yan, A. Hu, H. Liu, Q. Qian, J. Zhang, and F. Huang, "mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 13040–13051, 2024. 65, 68

[226] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016. 70

[227] D. Vint, G. Di Caterina, J. Soraghan, R. Lamb, and D. Humphreys, "Analysis of deep learning architectures for turbulence mitigation in long-range imagery," in *Artificial Intelligence and Machine Learning in Defense Applications II*, vol. 11543, p. 1154303, International Society for Optics and Photonics, 2020. 74

[228] D. Voelz, "Sampled functions and the discrete fourier transform," *Computational fourier optics: a MATLAB tutorial*, 2011. 78

[229] R. J. Eckert, "Polar phase screens: a comparison with other methods of random phase screen generation," 2006. 84

[230] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009. 105, 166

[231] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1653–1660, 2014. 111

[232] H. Elwarfalli and R. C. Hardie, "Fifnet: a convolutional neural network for motion-based multiframe super-resolution using fusion of interpolated frames," *Computer Vision and Image Understanding*, vol. 202, p. 103097, 2021. 111

[233] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. Change Loy, "Edvr: Video restoration with enhanced deformable convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019. 113, 117, 118, 119, 121, 174

[234] C. Tian, Y. Xu, and W. Zuo, "Image denoising using deep cnn with batch renormalization," *Neural Networks*, vol. 121, pp. 461–473, 2020. 114, 115, 117, 222

[235] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2472–2481, 2018. 114, 115, 117, 126, 222

[236] R. Feng, J. Gu, Y. Qiao, and C. Dong, "Suppressing model overfitting for image super-resolution networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019. 114, 115, 117, 223

[237] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 286–301, 2018. 114, 115, 117, 118, 223

[238] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883, 2016. 115, 118

[239] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018. 118

[240] Y. Tian, Y. Zhang, Y. Fu, and C. Xu, "Tdan: Temporally-deformable alignment network for video super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3360–3369, 2020. 120

[241] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 694–711, Springer, 2016. 122

[242] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4161–4170, 2017. 126

[243] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016. 130, 178

[244] T. Wang, Y. Zhu, L. Jin, C. Luo, X. Chen, Y. Wu, Q. Wang, and M. Cai, "Decoupled attention network for text recognition," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 12216–12224, 2020. 130, 178

[245] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "Aster: An attentional scene text recognizer with flexible rectification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2035–2048, 2018. 130, 178

[246] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8934–8943, 2018. 162

[247] T. Karras, "A style-based generator architecture for generative adversarial networks," *arXiv preprint arXiv:1812.04948*, 2019. 166

[248] A. Mishra, K. Alahari, and C. Jawahar, "Scene text recognition using higher order language priors," in *BMVC-British machine vision conference*, BMVA, 2012. 166