“Atmosphere Turbulence Mitigation”
Semester Project Report

- EP20BTECH11019 (Rahhul J)

Overview:

In this report, we will discuss about the things that were done during my Semester Project on the topic: “Atmospheric Turbulence Mitigation”. Here we will have a literature review review of: “**Single Frame Atmospheric Turbulence Mitigation: A Benchmark Study and A New Physics-Inspired Transformer Model**” paper authored by: Zhiyuan Mao, Ajay Jaiswal, Zhangyang Wang, and Stanley H. Chan along with the code for the TurbNet Model that was proposed. Then we will have a brief look at the model code of the Fully Connected Multi-Layer Perceptron Network designed for the objective to remove static turbulence from abberated images.

Literature Review:

1 Introduction

A part of the project is a literature review of: “**Single Frame Atmospheric Turbulence Mitigation: A Benchmark Study and A New Physics-Inspired Transformer Model**” paper authored by: Zhiyuan Mao, Ajay Jaiswal, Zhangyang Wang, and Stanley H. Chan. Image processing algorithms for mitigating the atmospheric turbulence effect have been studied for decades. However, many of them have limitations that prohibit them from being launched to practical systems:

1) Many of the existing algorithms are based on the principle of *lucky imaging* that requires multiple input frames. These methods often have a strong assumption that both the camera and the moving objects are static, which can easily become invalid in many real applications.

2) The conventional algorithms are often computationally expensive, making them unsuitable for processing large-scale datasets to meet the need of the latest computer vision systems.

3) Existing deep learning solutions are not utilizing the physics of the turbulence. Many of them are also tailored to recovering faces instead of generic scenes. The generalization is therefore a question.

4) The algorithms may not be properly evaluated due to the absence of a widely accepted real large-scale benchmarking dataset.

To articulate the aforementioned challenges, in this project three contributions were made:

1. A comprehensive benchmark evaluation of deep-learning based image restoration algorithms through atmospheric turbulence was presented. A sophisticated physics-grounded simulator was tuned to generate a large-scale dataset, covering a broad variety of atmospheric turbulence effects.

The highly realistic and diverse dataset leads to exposing shortages of current turbulence mitigation algorithms.

2. Realizing the existing algorithms' limitations, a novel physics inspired turbulence restoration model was was introduced, termed *TurbNet*. Built on a transformer backbone, *TurbNet* features a modularized design that targets modeling the spatial adaptivity and long-range dynamics of turbulence effects, plus a self-supervised consistency loss.
3. A variety of evaluation regimes were presented and two large-scale real world turbulence *testing* datasets were collected, one using the heat chamber for classical objective evaluation (e.g., PSNR and SSIM), and one using real long-range camera for optical text recognition as a semantic "proxy" task.

2 Related Works

The atmospheric turbulence mitigation methods have been studied by the optics and vision community for decades. To reconstruct a turbulence degraded image, conventional algorithms often adopt the multi-frame image reconstruction strategy. The key idea is called "lucky imaging", where the geometric distortion is first removed using image registration or optical flow techniques. Sharper regions are then extracted from the aligned frames to form a lucky frame. A final blind deconvolution is usually needed to remove any residue blur. These methods are usually very computationally expensive.

Recent deep learning methods adopt more dynamic strategies. Li et al. propose to treat the distortion removal as an unsupervised training step. While it can effectively remove the geometric distortions induced by atmospheric turbulence, its computational cost is comparable to conventional methods, as it needs to repeat the training step for each input image. There are also several works that focus on specific types of images, such as face restoration. They are usually based on a simplified assumption on atmospheric turbulence where they assume the blur to be spatially invariant. Such assumption cannot extend to general scene reconstruction, where the observed blur can be highly spatially varying due to a wide field of view.

Despite recent advances in turbulence mitigation algorithms, there is a very limited amount of publicly available datasets for atmospheric turbulence. The scale of these datasets is not suitable for evaluating modern learning-based methods.

Due to the nature of the problem, it is very difficult to obtain aligned clean and corrupted image pairs. Existing algorithms are all trained with synthetic data. The computationally least expensive synthesis technique is based on the random pixel displacement + blur model. Our data synthesis scheme is based on the *P2S* model.

3 Restoration Model

3.1 Problem Setting and Motivation

Consider a clean image **I** in the object plane that travels through the turbulence to the image plane. Following the classical split-step wave-propagation equation, the resulting image $\tilde{\boldsymbol{I}}$ is constructed through a sequence of operations in the phase domain:

$$\mathbf{I} \rightarrow \text{Fresnel} \rightarrow \text{Kolmogorov} \rightarrow \cdots \text{Fresnel} \rightarrow \text{Kolmogorov} \rightarrow \widetilde{\mathbf{I}}, \tag{1}$$

where “Fresnel” represents the wave propagation step by the Fresnel diffraction, and “Kolmogorov” represents the phase distortion due to the Kolmogorov power spectral density.

Certainly, Eqn. 1 is implementable as a forward equation (i.e, for simulation) but it is nearly impossible to be used for solving an inverse problem. To mitigate this modelling difficulty, one computationally efficient approach is to approximate the turbulence as a composition of two processes:

$$\widetilde{\mathbf{I}} = \Big(\underbrace{\mathcal{H}}_{\text{blur}} \circ \underbrace{\mathcal{G}}_{\text{geometric}} \Big)(\mathbf{I}) + \mathbf{N}, \tag{2}$$

where H is a convolution matrix representing the spatially *varying* blur, and G is a mapping representing the geometric pixel displacement (known as the tilt). The variable **N** denotes the additive noise / model residue in approximating Eqn. 1 with a simplified model. The operation “$\circ$” means the functional composition. That is, we first apply G to **I** and then apply H to the resulting image.

The simultaneous presence of H and G in Eqn. 2 makes the problem hard. If there is only H, the problem is a simple deblurring. If there is only G, the problem is a simple geometric unwrapping. Generic deep-learning models such as [36,23] adopt network architectures for classical restoration problems based on conventional CNNs, which are developed for one type of distortion. Effective, their models treat the problem as

$$\tilde{\boldsymbol{I}} = \mathsf{T}(\mathbf{I}) + \mathbf{N}, \tag{3}$$

where $\mathsf{T} = \mathsf{G} \circ \mathsf{H}$ is the overall turbulence operator. Without looking into how T is constructed, existing methods directly train a generic restoration network by feeding it with noisy-clean training pairs. Since there is no physics involved in this generic procedure, the generalization is often poor.

Contrary to previous methods, in the paper, they proposed to jointly estimate the physical degradation model T of turbulence along with reconstruction of clean image from the degraded input $\tilde{\boldsymbol{I}}$. Such formulation explicitly forces our model to focus on learning a generic turbulence degradation operator independent of image contents, along with the reconstruction operation to generate clean output. Moreover, the network training is assisted by high-

quality, large-scale, and physics-motivated synthetic training data to better learn the key characteristics of the atmospheric turbulence.

3.2 Model Architecture

Turbulence and limitation of CNNs: CNNs have been *de facto* choice by most of the previous image restoration algorithms, yet they are limited by two primary issues:

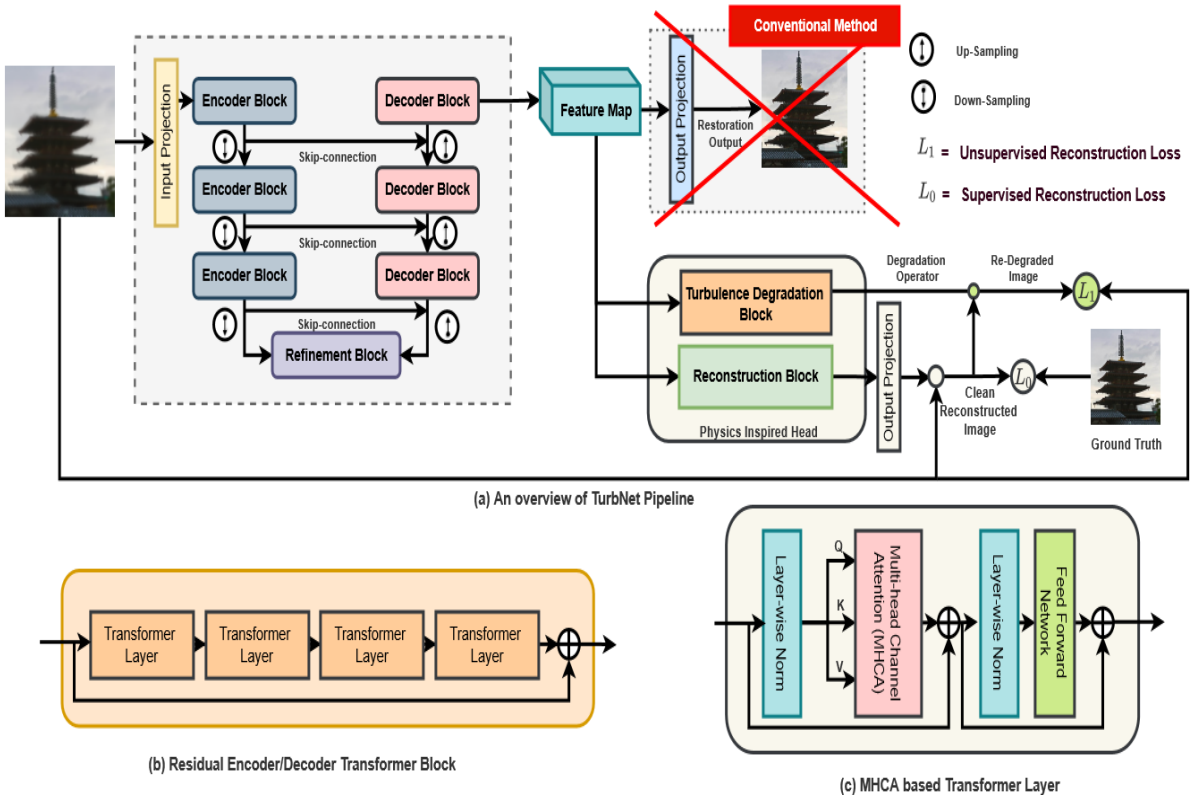


Fig.1. Architecture of the proposed method. (a) The overall architecture consists: (i) a transformer to pull the spatially dynamical features from the scene; (ii) instead of directly constructing the image, we introduce a physics-inspired model to estimate the turbulence while reconstructing the image. (b) The structure of the residual encoder/decoder transformer block. (c) The details of each transformer layer.

1) The convolutional filters cannot adapt to image content during inference due to their static weights.

2) The local receptive fields cannot model the long-range pixel dependencies.

Previous restoration methods treat turbulence restoration as a regression problem using CNNs but ignore the fact that turbulence is highly location adaptive and should not be represented as static fixed kernel applied to all locations. It is not difficult to see that applying static weight convolutions to regions with drastically different distortions will lead to sub-optimal performance.

Leveraging the capability of multi-head self-attention, we propose the *TurbNet*, a transformer-based end-to-end network for restoring turbulence degraded images. Transformer-based architecture allows the creation of input-adaptive and location-adaptive filtering effect using *key, query,* and *weight*, where *key* and *query* decide content-adaptivity while *weight* brings location-adaptivity. The design, as shown in Figure 1, is composed of several key building blocks:

Transformer Backbone: The proposed network consists of a transformer based backbone that has the flexibility of constructing an input-adaptive and location-adaptive unique kernel to model spatially- and instance-varying turbulence effect. TurbNet adopts a U-shape encoder-decoder architecture due to its hierarchical multi-scale representation while remaining computationally efficient. Our backbone consists of three modules: input projection, deep encoder and decoder module. Input project module uses convolution layers to extract low frequency information and induces dose of convolutional inductive bias in early stage and improves representation learning ability of transformer blocks. Deep encoder and decoder modules are mainly composed of a sequential cascade of Multi-head channel attention (MHCA) based transformer layers. Compared to prevalent CNN-based turbulence mitigation models, this design allows content-based interactions between image content and attention weights, which can be interpreted as spatially varying convolution.

The primary challenge of applying conventional transformer blocks for image restoration task comes from the quadratic growth of key-query dot product interactions, i.e., $O(W^2H^2)$, for images with $W \times H$ pixels. To alleviate this issue, we adopt the idea of applying self-attention across channels instead of spatial dimension [37], and compute cross-covariance across channels generating attention map. Given *query* (**Q**), *key* (**K**), and *value* (**V**), we reshape **Q** and **K** such that their dot-product generates a transposed-attention map $\mathbf{A} \in R^{C\times C}$, instead of conventional $R^{HW\times HW}$ [5]. Overall, the MHCA can be summarized as:

$$\mathbf{X}' = \mathbf{W_p}\,\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) + \mathbf{X} \tag{4}$$

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{V} \cdot \text{softmax}\left\{\frac{\mathbf{K}\cdot\mathbf{Q}}{\alpha}\right\} \tag{5}$$

where $\mathbf{X}'$ and $\mathbf{X}$ are input and output feature maps, $\mathbf{W_p}^{(\cdot)}$ is the 1 × 1 point-wise convolution, and α is a learnable scaling parameter to control the magnitude of ($\mathbf{K} \cdot \mathbf{Q}$) before applying softmax.

Image Reconstruction Block: To further enhance deep features generated by the transformer backbone, TurbNet uses the reconstruction block. The primary job of the reconstruction block is to take deep features corresponding to turbulence degraded input image $\tilde{I}$ by the transformer backbone, further enrich it at high spatial resolution by encoding information from spatially neighbouring pixel positions. Next, the enriched features pass through an output projection module with 3 × 3 convolution layers to project it back low dimension feature map corresponding to the reconstructed clean image **J**.

Precisely, the work of Reconstruction module can be summarized as:

$$\underset{\substack{\text{Deep Features of degraded} \\ \text{Input Image } \tilde{I}}}{\mathbf{F}\_\tilde{I}} \rightarrow \text{Reconstruction Module} \rightarrow \underset{\substack{\text{Reconstructed Clean} \\ \text{Output Image}}}{\mathbf{J}\_\tilde{I}} \quad (6)$$

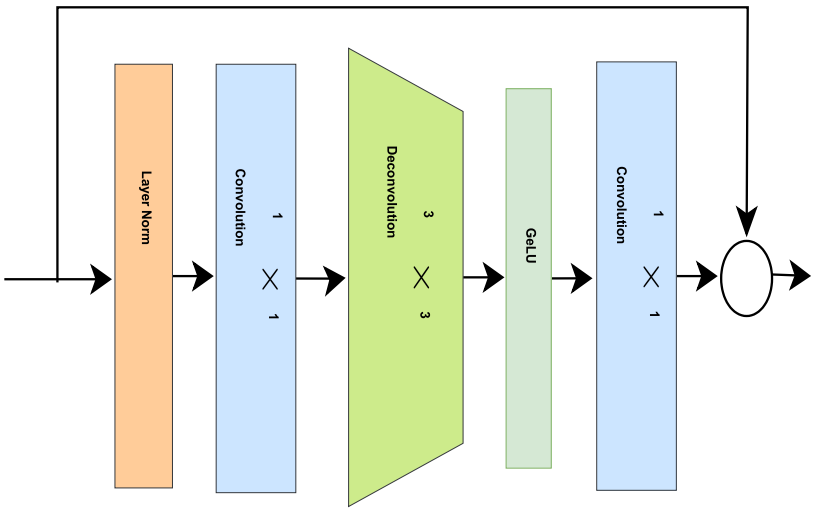


Fig.2. Locally-Enhanced Feed Forward Network (LoFFN) used in the image reconstruction block and the turbulence degradation block.

Turbulence Degradation Block: In TurbNet, the turbulence degradation module learns the physical turbulence degradation operator T from the input synthetic training data. The primary job of turbulence degradation module is to take clean reconstructed image $\mathbf{J}\_\tilde{I}$ corresponding to degraded input image $\tilde{I}$,

apply the learned degradation operator T , to construct back the **re-degraded** input image $\tilde{\boldsymbol{I}}_{\mathrm{T}}$. This formulation enriches the training set by incorporating additional latent degradation images ($\tilde{\boldsymbol{I}}_{\mathrm{T}}$), in addition to synthesized degraded images ($\tilde{\boldsymbol{I}}$), during the training process. Additionally, this module facilitates selfsupervised learning without the availability of ground truth. The architecture of this module is the same as Image Reconstruction Block with LoFFN.

Precisely, the work of Degradation Block can be summarized as:

$$\underbrace{\mathbf{J}_{\mathbf{el}}}_{\substack{\text{Reconstructed Clean} \\ \text{Output Image}}} \rightarrow \text{Degradation Operator}^{\mathcal{T}(\cdot)} \rightarrow \underbrace{\widetilde{\mathbf{I}}_{\mathcal{T}}}_{\substack{\text{Re-degraded} \\ \text{Output Image}}} \tag{7}$$

Loss Function: TurbNet optimization requires the joint optimization of reconstruction operation and the turbulance degradation operation. Given the synthetic training pair of degraded input $\tilde{\boldsymbol{I}}$, and corresponding ground truth image **I**, following two losses are formulated:

$$\underbrace{\mathcal{L}_0}_{\text{Supervised Reconstruction Loss}} = ||\mathbf{J}_{\mathbf{el}} - \mathbf{I}||_1 \tag{8}$$

$$\underbrace{\mathrm{L}_1}_{\text{Self-supervised Reconstruction Loss}} = ||\widetilde{\mathbf{I}}_{\mathcal{T}} - \widetilde{\mathbf{I}}||_1 \tag{9}$$

where, L_0 is responsible for constructing a clean image $\mathbf{J}\_\tilde{\boldsymbol{I}}$ given the degraded input image $\tilde{\boldsymbol{I}}$, L_1 helps to ensure degradation operator T can reconstruct the original the original input $\tilde{\boldsymbol{I}}$ from the reconstructed clean image $\mathbf{J}\_\tilde{\boldsymbol{I}}$.

Eventually, the overall loss L to train TurbNet can be summarized as:

$$\mathrm{L} = \alpha \times \mathrm{L}_0 + (1 - \alpha) \times \mathrm{L}_1 \tag{10}$$

Overall Pipeline As shown in Figure 1(a), TurbNet utilizes a U-shape architecture built upon transformer blocks to extract deep image features.An initial convolution-based input projection is used to project the input image to higher dimensional feature space, which can lead to more stable optimization and better results. After obtaining the feature maps, TurbNet jointly learns the turbulence degradation operator (T) along with the reconstructed image ($\mathbf{J}\_\tilde{\boldsymbol{I}}$), in contrary to general image restoration methods [32,19,2,37] that directly reconstruct the clean image. This design facilitates spatial adaptivity and long-range dynamics of turbulence effects, plus a self-supervised consistency loss.

Synthetic-to-Real Generalization: With a pre-trained TurbNet model M(·) using the synthetic data, TurbNet design allows an effective way of generalizing M(·) on unseen real data (if required) with the help of degradation operator T (·) in a self-supervised way. Starting from model M(·), a generalization dataset is created by incorporating unlabelled real data with the synthetic data to fine-tune M(·). For input images with no ground truth, M(·) is optimized using Equation (9), while for input images from labeled synthetic data M(·) is optimized using Equation (8, and 9). Note that we incorporate synthetic data into the fine-tuning process to mitigate the issue of catastrophic forgetting during generalization.

4 Large-Scale Training and Testing Datasets

4.1 Training Data: Synthetic Data Generating Scheme

Training a deep neural network requires data, but the real clean-noisy pair of turbulence is nearly impossible to collect. A more feasible approach here is to leverage a powerful turbulence simulator to synthesize the turbulence effects.

The deep learning model generates the geometric distortions by repeatedly smoothing a set of random spikes, and the blur is assumed to be spatially invariant Gaussian. For more complex scenarios, such a simplified model will fail to capture two key phenomena that could cause the training of the network to fail:

(1) The instantaneous distortion of the turbulence can vary significantly from one observation to another even if the turbulence parameters are fixed. See Figure 3(a) for an illustration from a real data.

(2) Within the same image, the distortions are spatially varying. See Figure 3(b).

In order to capture these phenomena, we adopt an advanced simulator to synthesize a large-scale *training* dataset for atmospheric turbulence. The clean data used by the simulator is the *Places* dataset. A total of 50,000 images are generated, and the turbulence parameters are configured to cover a wide range of conditions.

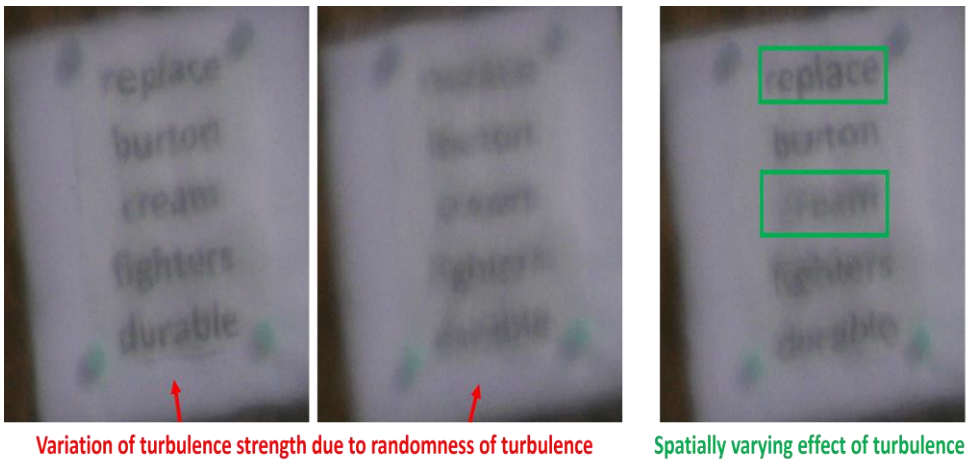


Fig.3. Key turbulence effects requiring attention while designing synthetic dataset.

4.2 Testing Data: Heat Chamber and Text Datasets

The real benchmarking dataset consists of two parts: the *Heat Chamber Dataset* and the *Turbulent Text Dataset*. Although the paper focuses on single frame restoration, both the benchmarking datasets contain 100 static turbulence degraded frames for each scene.

Heat Chamber Dataset. The *Heat Chamber Dataset* is collected by heating the air along the imaging path to artificially create a stronger turbulence effect. The setup for collecting the heat chamber dataset is shown in 4. Turbulence free ground truth images can be obtained by shutting down the heat source. The images are displayed on a screen placed 20 meters away from the camera.

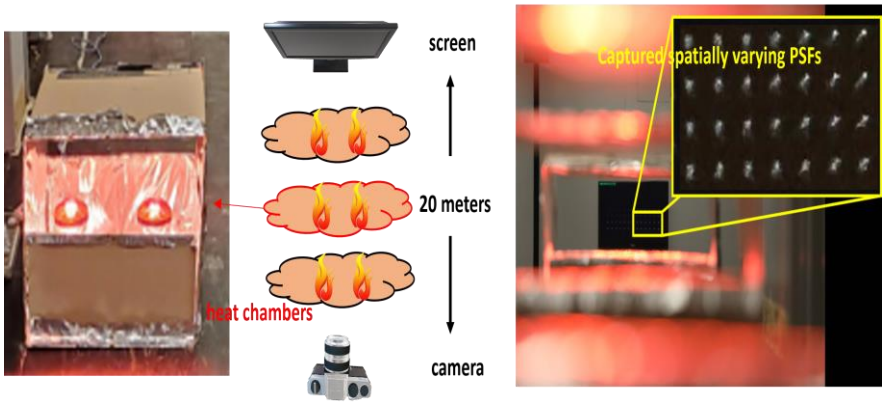


Fig.4. The setup of heat chamber data collection. We evenly placed three heat chambers along the imaging path. Our dataset captures better spatially varying effect.

We remark that our data has a clear improvement: we use a long path and more evenly distributed heat so that the turbulence effect is closer to the true long-range effect. The captured images have a better aniso-planatic (spatially varying) effect such that an almost distortion-free frame is less likely to occur compared with the dataset in [10,1]. In addition, our dataset is much large in scale. It contains 2400 different images, which allows for a better evaluation of the learning-based model. Sample images of the *Heat Chamber Dataset* can be found in Figure 5.

Turbulence Text Dataset. Due to the nature of the problem, it is extremely difficult, if not impossible, to capture ground truth clean images in truly long-range settings.

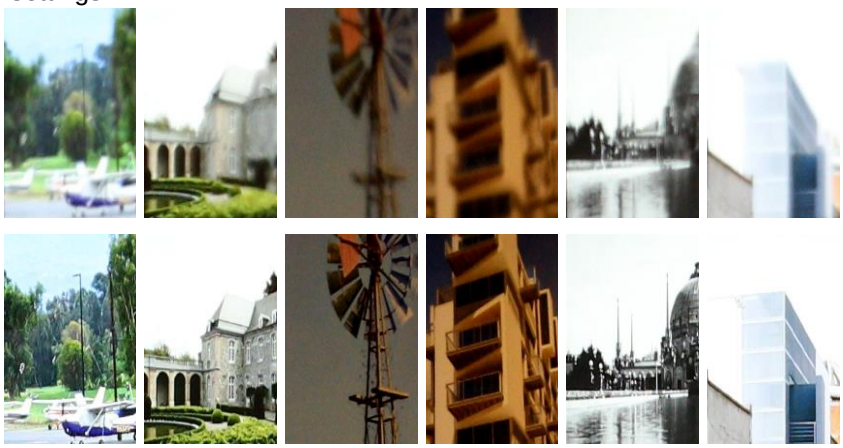


Fig.5. Sample turbulence degraded images (top) and corresponding ground truth (bottom) from our *Heat Chamber Dataset*. The D/r_0 is estimated to be around 3.

Therefore, they adopted the idea of using the performance of high-level vision task as an evaluation metric for image restoration. Specifically, we calculate the detection ratio and longest common subsequence on the output of an OCR algorithm as the evaluation metrics.

There are several advantages of using text recognition:

1) The degradation induced by atmospheric turbulence, the geometric distortion and the loss of resolution, can be directly reflected by the text patterns. Both

types of degradation need to be removed for the OCR algorithms to perform well.

2) The OCR is a mature application. The selected algorithms should be able to recognize the text patterns as long as the turbulence is removed. Other factors such as the domain gap between the training and testing data will not affect the evaluation procedure as much as other high-level vision tasks.

3) An important factor to consider when designing the dataset is whether the difficulty of the task is appropriate. The dataset should neither be too difficult such that the recognition rate cannot be improved by the restoration algorithms nor too easy making all algorithms perform similarly. We can easily adjust the font size and contrast of text patterns to obtain a proper difficulty level.

The *Turbulence Text Dataset* consists of 100 scenes, where each scene contains 5 text sequences. Each scene has 100 static frames. It can be assumed that there is no camera and object motion within the scene, and the observed blur is caused by atmospheric turbulence. The text patterns come in three different scales, which adds variety to the dataset. We also provide labels to crop the individual text patterns from the images. Sample images from the dataset are shown in Figure 6.

5 Experiment Results

Implementation Details: TurbNet uses a 4-staged symmetric encoder-decoder architecture, where stage 1, 2, 3, and 4 consist of 4, 6, 6, and 8 MHCA-based transformer layers respectively. The Reconstruction block and Turbulence Degradation block consist of 4 MHCA-transformer layers enhanced with LoFFN. TurbNet is trained using 50,000 synthetic dataset generated using a physics-based

(a) Data collection site. (b) Sample collected images

Fig.6. Data collection site of the *Turbulence Text Dataset*. The distance between the camera and the target is 300 meters. The D/r_0 is estimated to be in range of 2.5 to 4 (varies due to the temperature change during the collection process). The collected text patterns are in 3 different scales.

Table 1. Performance comparison of state-of-art restoration baselines with respect to TurbNet on synthetic and *Heat Chamber* dataset.

	TDRN[35]	MTRNN[24]	MPRNet[38]	Uformer[32]	Restormer[37]	**TurbNet**
			Synthetic Dataset			
PSNR	21.35	21.95	21.78	22.03	22.29	**22.76**
SSIM	0.6228	0.6384	0.6410	0.6686	0.6719	**0.6842**
			HeatChamber Dataset			
PSNR	18.42	18.12	18.68	19.12	19.01	**19.76**
SSIM	0.6424	0.6379	0.6577	0.6840	0.6857	**0.6934**

stimulator and MIT Places dataset while synthetic evaluation results are generated on 5,000 synthetic images. Due to resource constraint, our synthetic training uses a batch size of 8 with Adam optimizer. We start our training with learning rate of $1e-4$, and use the cosine annealing scheduler to gradually decrease the learning rate over the span of 50 epcohs. During training, to modulate between the loss Equation 8 and 9, we have use α to be 0.9.

5.1 Synthetic and Heat Chamber Dataset Results

We first conduct an experiment on a synthetic testing dataset generated with the same distribution as testing data. In Figure 7, we show a qualitative comparison between our restored images with ground truth. It can be seen that

our results are accurately reconstructed with the assist from estimated turbulence map.

It can be observed that the transformer-based methods generally perform better than the CNN-based methods due to their ability to adapt dynamically to the distortions. The proposed method achieves authentic reconstruction due to its ability to explicitly model the atmospheric turbulence distortion. Table 1 presents the quantitative evaluation of TurbNet wrt. other baselines. TurbNet achieves the best results in both PSNR and SSIM. Note that Uformer[32], and Restomer[37] (designed for classical restoration problems like deblurring, deraining, etc.) uses transformer-based encoder decoder architecture, but their performance is significantly low than TurbNet, which validates the importance of the decoupled (reconstruction and degradation estimation) design.

5.2 Turbulence Text Dataset Results

Evaluation Method: In order to evaluate the performance of TurbNet on our real-world turbulence text dataset, publicly available OCR detection and recognition algorithms are used. We propose the following two evaluation metrics - Average Word Detection Ratio (**AWDR**), and Average Detected Longest Common Subsequence (**AD − LCS**) defined as follows:

$$\textbf{AWDR} = \frac{\sum_{Scene=1}^{N} \frac{\text{Word Detected}_{scene}}{\text{Word Count}}}{N}_{scene}, \tag{11}$$

$$\textbf{AD} - \textbf{LCS} = \frac{\sum_{Scene=1}^{N} \sum_{Word=1}^{K} \mathcal{LCS}(DetectedString, TrueString)}{N}, \tag{12}$$

where LCS represents the Longest Common Subsequence, *TrueString* represents the ground truth sequence of characters corresponding to a word *i* in the image, *DetectedString* represents a sequence of characters recognized by OCR algorithms for word *i*, and *N* is the total number of scenes in the test dataset.

Table 2. Performance comparison of state-of-art restoration baselines with respect to TurbNet on our *Turbulence Text Dataset*.

Raw Input	TDRN[35]	MTRNN[24]	MPRNet[38]	Restormer[37]	**TurbNet**

AWDR	0.623	0.617	0.642	0.633	0.702	**0.758**
AD-LCS	5.076	5.011	5.609	5.374	6.226	**7.314**

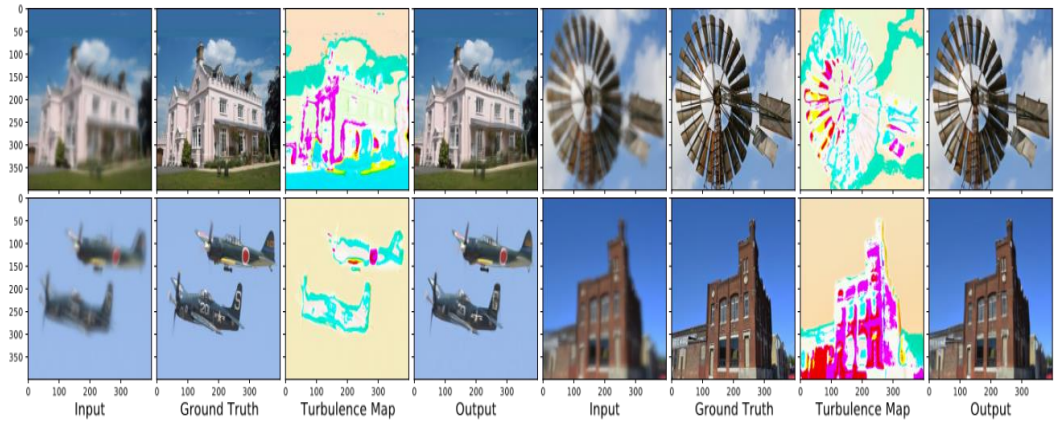


Fig.7. Qualitative Performance comparison of TurbNet wrt. the ground truth.

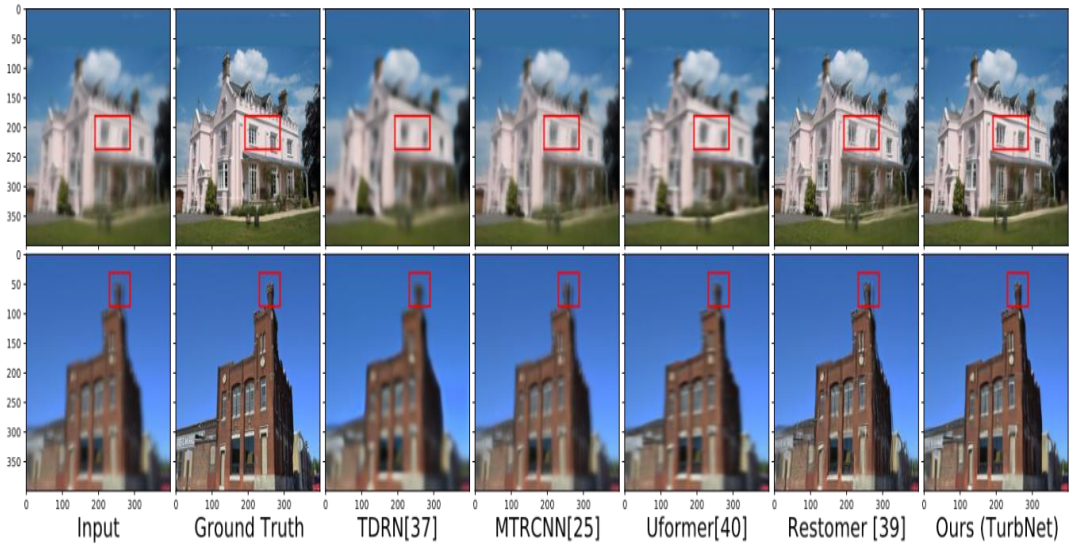


Fig.8. Qualitative Performance comparison of TurbNet wrt. other SOTA methods.

Discussion: Figure 9 represents the performance of OCR on the real turbulence impacted images and images restored by TurbNet. It is evident that our restoration model significantly helps in improving the OCR performance by identifying comparatively more words with higher confidence. Table 2 presents the performance gain by TurbNet over the real turbulence degraded text images and their restored version by various state-of-the-art methods. OCR algorithms achieve massive improvements of +0.135 (AWDR) and +2.238 (AD-LCS) when

used on images restored by TurbNet compared to being used directly on real images from our proposed test dataset.

5.3 Experimental Validity of the Proposed Model

We conduct two additional experiments to validate the proposed model. The first experiment is an ablation study, where we demonstrate the impact of replacing transformer as feature encoder with U-Net [26] and removing the turbulence map estimation part. The result is reported in 3, where we observe a significant performance drop in both cases. The second experiment is to prove the effectiveness of the extracted turbulence map. We extract a turbulence map from a simulated frame and apply the map back to the ground-truth image. We calculate the PSNR of this re-corrupted image w.r.t. the original turbulence frame. We tested on 10K turbulence frames and the average PSNR is **39.89 dB**, which is a strong evidence that our turbulence map can effectively extract the turbulence information embedded in the distorted frames. A visualization of the experiment can be found in Figure 10.

Table 3. Ablation on Heat Chamber Dataset

Model type	PSNR	SSIM
TurbNet [Ours]	19.76	0.6934
TurbNet - Turbulance Map	19.03 (↓)	0.6852 (↓)
TurbNet - Transformer	18.62 (↓)	0.6481 (↓)

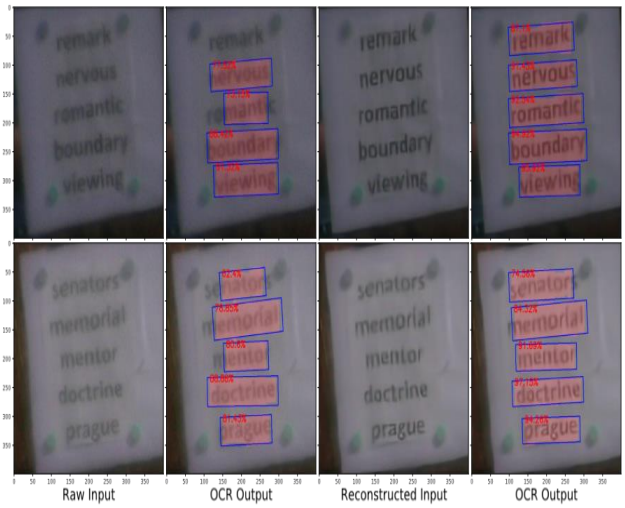


Fig. 9. OCR performance of our reconstruction algorithm for *Turbulance Text Dataset*

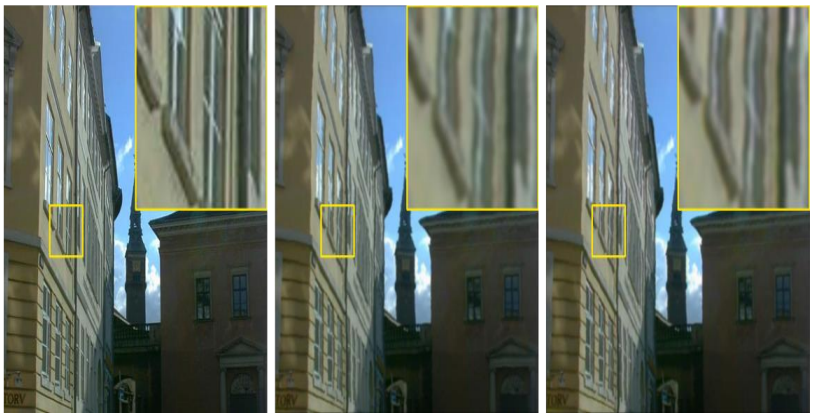


Fig.10. Validation of our turbulence map. Left: groundtruth. Middle: original turbulence frame. Right: groundtruth re-corrupted with the extracted turbulence map.

6 Conclusions

In this work, identifying the short-come of existing image restoration algorithms, A novel physics-inspired turbulence restoration model (TurbNet) based on transformer architecture to model spatial adaptivity and long-term dynamics of turbulence effect was proposed. A synthetic data generation scheme for tuning a sophisticated physics-grounded simulator to generate a large-scale dataset was presented, covering a broad variety of atmospheric turbulence effects. Additionally, two new large-scale testing datasets that allow for evaluation with classical objective metrics and a new task-driven metric with optical text

recognition was introduced. The comprehensive evaluation on realistic and diverse datasets leads to exposing limitations of existing methods and the effectiveness of TurbNet.

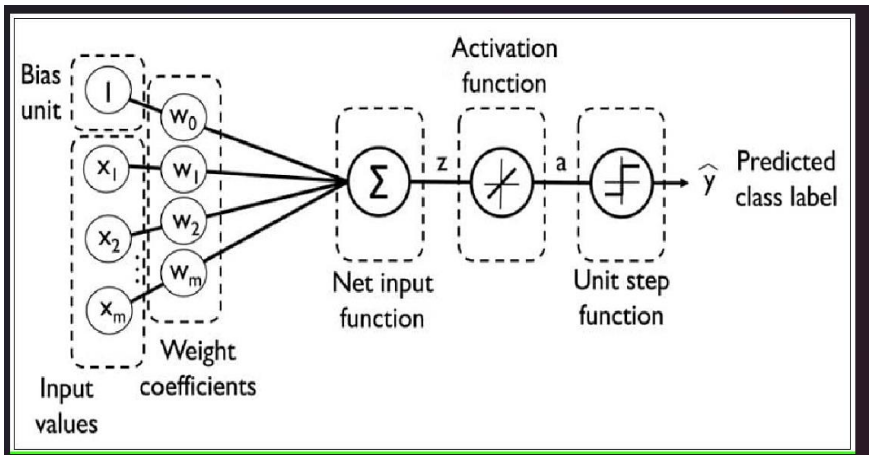
7 TurbNet Source Code

The source code for the propsed TurbNet model can be found in the below Github link: https://github.com/rahhul17/TurbNet

Fully Connected Multi-Layer Perceptron Network:

In the next part of the project, we are going to see about the Multi-layer Perceptron Network that was designed for the objective of atmospheric turbulence mitigation. The Code for the same can be found in the below Github Link: https://github.com/rahhul17/ao_project/blob/main/ao_nn2.py

The Image below depicts the typical structure of a fully-connected Multi-layer Perceptron:



Description:

The model in the previous code is a neural network architecture for adaptive optics. It consists of four convolutional layers and three fully connected layers.

The first convolutional layer takes a single-channel image as input and applies 16 filters of size 3x3 to it, with padding of 1. The output of this layer is then passed through a max-pooling layer with a kernel size of 2x2 and a stride of 2. The second convolutional layer takes the output of the first layer as input and applies 32 filters of size 3x3 to it, with padding of 1. The output of this layer is then passed through another max-pooling layer.

The third convolutional layer takes the output of the second layer as input and applies 64 filters of size 3x3 to it, with padding of 1. The output of this layer is then passed through another max-pooling layer. The fourth convolutional layer takes the output of the third layer as input and applies 128 filters of size 3x3 to it, with padding of 1. The output of this layer is then passed through another max-pooling layer. The output of the fourth convolutional layer is flattened and passed through two fully connected layers with 512 and 256 neurons, respectively. The output of the last fully connected layer is a 2-dimensional vector, which represents the x and y position of the adaptive optics system.

The model uses ReLU activation function after each convolutional and fully connected layer, and applies dropout with a probability of 0.5 after the first two fully connected layers to reduce overfitting. The model is optimized using the Adam optimizer with a learning rate of 0.001, and the mean squared error (MSE) loss function is used as the loss metric.