Physics Informed and Data Driven Simulation of Underwater Images via Residual Learning

Tanmoy Mondal[1], Ricardo Mendoza[2], and Lucas Drumetz[1]

[1] TOMS Team, IMT Atlantique, Brest, France,
tanmoy.besu@gmail.com, lucas.drumetz@imt-atlantique.fr
[2] Cervval, Brest, France,
mendoza@cervval.com

Abstract. In general, underwater images suffer from color distortion and low contrast, because light is attenuated and backscattered as it propagates through water (differently depending on wavelength and on the properties of the water body). An existing simple degradation model (similar to atmospheric image "hazing" effects), though helpful, is not sufficient to properly represent the underwater image degradation because there are unaccounted for and non-measurable factors e.g. scattering of light due to turbidity of water, reflective characteristics of turbid medium etc. We propose a deep learning-based architecture to automatically simulate the underwater effects where only a dehazing-like image formation equation is known to the network, and the additional degradation due to the other unknown factors if inferred in a data-driven way. We only use RGB images (because in real-time scenario depth image is not available) to estimate the depth image.
For testing, we have proposed (due to the lack of real underwater image datasets) a complex image formation model/equation to manually generate images that resemble real underwater images (used as ground truth). However, only the classical image formation equation (the one used for image dehazing) is informed to the network. This mimics the fact that in a real scenario, the physics are never completely known and only simplified models are known. Thanks to the ground truth, generated by a complex image formation equation, we could successfully perform a qualitative and quantitative evaluation of proposed technique, compared to other purely data driven approaches. For code and dataset, see: https://github.com/anoynymREVIEW/underwater_simulation.git

Keywords: Under water image, Dehazing, Denoising, Image Simulation, Image-to-Image translation, Encoder-Decoder, DenseNet, Pix2Pix, Cycle GAN.

1 Introduction

Underwater images lack contrast and contain a different color palette from usual natural images. This occurs because the light spectrum is selectively absorbed and scattered (mainly due to the floating particles in the water) during the propagation of light in water. The attenuation of light highly depends on the wavelength, which varies with respect to the water type, depth and the distance which light has to travel to illuminate the object [1]. Underwater images also depend on the $3D$ structure of the scene and floating particles in the water, which makes it very difficult to model the underwater scattering

phenomena. The wavelength-dependent attenuation of light causes color distortions and are directly related to the objects' distances from the source of light. Furthermore, light scattering introduces an additional factor into the image which inherently decreases the image's contrast. Light scattering is directly related to the object's distance from the light's source, which explains why these phenomena cannot be easily globally corrected [2]. Moreover, the attenuation parameters of the water medium are highly affected by seasonal, geographical and climate variations. These variations of attenuation parameters were categorized into 10 categories by Jerlov [3].

Furthermore, underwater image formation also gets affected by the scattering of light due to reflective characteristics of the turbid medium, among others. More importantly, these factors are not measurable and difficult to incorporate within a mathematical model. In this work, we propose an end-to-end deep learning-based physics-informed model to simulate the underwater effects. A classical image formation equation is hard coded into the network, and it estimates additional non-measurable and complex factors which influence the underwater image degradation. We have only used clean *RGB* atmospheric images and have estimated the depth image (the depth values of each image pixel) from it. Because in a real scenario, we do not have the access to each pixel's depth values (because we need to use *RGB-D* cameras, e.g. Microsoft Kinect, since classical *RGB-D* cameras cannot be used for underwater imaging). The estimated depth image is fed to the physical model part of the network.

Due to the inherent difficulty of obtaining pairs of real-world clean/degraded images in an underwater context, we propose in addition a complex image formation model/equation to manually generate images that resemble real-world underwater images. The generated images are used as ground truth for our experiments. This work is a proof of concept, where the objective is to simulate images as close as possible to the observed data (which we manually generate using our proposed physical model, mentioned in Equation (6)), while capturing unaccounted and unmeasurable physical effects in a data-driven manner. A rich dataset of clean-degraded image pairs is created to train a neural network model that will be used as a simulator to generate varied underwater images, parameterized by a few user given parameters.

This way, once trained, our model can be effectively generate realistic rare underwater images and provide an efficient physically explainable emulator. To the best of our knowledge, there are no other research works on the simulation of underwater images. The proposed physics-data-driven method to simulate under water image degradation effects using a deep neural network is a novel technique. The contribution of this work are as follows: *i)* We propose a complex image formation model/equation to manually generate images that resemble real underwater images (see section 3) which is used as ground truth. *ii)* Then we propose a deep neural network based architecture to simulate the complex under water imaging effect by informing the network about the classical image degradation model (see section 4), to make it interpretable and able to capture missing degradation in a data-driven way. We have further analyzed the influence and effectiveness of each block on the overall performance of the network, and compared it to simulators obtained from a number of other data-driven deep learning models.

2 Related Work

In this section, we discuss the literature related to the correction of haze-related degradation of terrestrial images, which has a thin connection with underwater image degradations. In case of terrestrial images, in the presence of fog, haze or turbulence, the transmitting light gets diffracted and scattered while passing through the atmosphere [4]. Many image dehazing techniques in the literature [5], [6], [7], [8], [9], [10] take as input a single hazy image and estimate the unknown distance map, from which the clear image of the scene is generated. Researchers have proposed several priors in order to solve the ill-posed inverse problem. It is assumed that transmission is color independent for terrestrial images but that is not the case for underwater images. One of such priors is the so-called "Dark Channel Prior (DCP)" [6] [7] which assumes that within small image patches, at least one pixel has a low value in some color channel. This minimal value is used to estimate the distance.

This concept has been widely used in the processing of underwater images [11], [12], [13], [14], [15]. Although, it works well in case of terrestrial images, the same underlying assumption does not hold in many underwater scenarios. For example, a bright sand foreground has high values in all color channels and is often mistaken to have low transmission despite being close to the camera. Because of these reasons, [16] proposed a depth estimation method for underwater scenes which is based on image blurriness and light absorption. Although such prior is physically valid, it has limited efficiency in texture-less areas. There are a few more works in the literature which focus on perceptually pleasing results e.g. [17], [18], [19] but have not shown color consistency which is required for scientific measurements. There have been only few number of attempts of using deep neural networks [20], [21], [22], [23] for the restoration of terrestrial haze images and even less for underwater images. In [24], authors estimate the ambient light and transmission in underwater images by using a classical convolution neural network (CNN) architecture which is further used to dehaze underwater images.

The simulation of underwater images has strong resemblance with the task of *Image-to-Image (I2I)* translation which aims to learn a mapping between different visual domains. This task is challenging for two main reasons. First, it is either difficult to collect aligned training image pairs (e.g. day scene $\leftrightarrow$ night scene), or they simply do not exist (e.g. artwork $\leftrightarrow$ real photo). Secondly, many such mappings are inherently multi-modal i.e. a single input may correspond to multiple possible outputs. Several techniques exists in the literature to address these issues. The *Pix2Pix* architecture [25] applies conditional generative adversarial networks to *I2I* translation problems by using paired image data. There are a number of recent works [26], [27], [28], [29], [25] which are based on the paired training data for learning *I2I* translation. These techniques produces a single output, conditioned on the given input image. To train with unpaired data, CycleGAN [30], DiscoGAN [31] leverage cycle consistency to regularize the training. Another set of unpaired I2I translation techniques either generate one (e.g. UNIT [28]) or many output images (e.g. MUNIT [32] and DRIT [33]) from a given input images, also by leveraging cycle consistency to regularize the training.

On the other hand, BicycleGAN [34] (only applicable to problems with paired training data) enforces a bijection mapping between the latent and target space to tackle the mode collapse problem. Contrary to all these approaches in the literature, the proposed

neural network architecture incorporates a physics-informed classical image degradation model to drive the network and helps him simulate more complex image degradation model whose formulation and related parameters are unknown.

3 Image Formation Model

3.1 Classical Simulation Model

In this section, we describe the widely used atmospheric image formation model developed in [35]. For each color channel $c \in \{R, G, B\}$, the image intensity at each pixel is composed of two components: the attenuated signal and the veiling light.

$$I_c(\mathbf{x}) = t_c(\mathbf{x})J_c(\mathbf{x}) + (1 - t_c(\mathbf{x})).A_c \tag{1}$$

where bold letters denotes vectors, $\mathbf{x}$ is the pixel coordinate, I_c is the acquired image value in the color channel c, t_c is the transmission of the color channel, and J_c is the object radiance or clean image. The global veiling-light/atmospheric light component A_c is the scene value in the areas with no objects ($t_c = 0, \forall c \in \{R, G, B\}$). The transmission depends on the object's distance $z(\mathbf{x})$ and the attenuation coefficient of the medium for each channel i.e. β_c:

$$t_c(\mathbf{x}) = \exp^{-\beta_c z(\mathbf{x})} \tag{2}$$

In the ocean, the attenuation of red colors can be an order of magnitude larger than the attenuation of blue and green [36]. Hence, contrary to the common assumption in single image dehazing, the transmission $\mathbf{t} = (t_R, t_G, t_B)$ is wavelength dependent.

The attenuation of light in underwater is not constant and varies with the change in geography, seasons and climate. The attenuation coefficient (β) is dependent on wavelength of various water types. Based on the water clarity, Jerlov [37] proposed a classification scheme for oceanic waters where open ocean waters are classified into class **I**, **IA**, **IB**, **II** and **III**. He also defined the water type **1** through **9** for coastal waters. Type **I** is the clearest and type **III** is the most turbid open ocean water. Similarly, for coastal water, type **1** is the clearest and type **9** is the most turbid. We use three attenuation coefficients: i.e. $\beta_R, \beta_G, \beta_B$ corresponding to RGB channels for our work.

3.2 Underwater Image Simulation Model

Obtaining a dataset of underwater images along with the ground truth information for depth, composition of veiling light and transmission coefficients is a challenging and expensive task. Here we describe our contribution to simulate underwater images, which is used to generate the ground truth of our underwater image dataset. This dataset aims to approximate certain complex underwater phenomena such as forward photon scattering in an absorbing medium [38] and turbidity in water due to colored dissolved matter [39], which inherently cause complementary image degradation (Fig. 1).

Our proposed model for underwater image simulation initiates from the classic atmospheric image formation model (see Equation 1) and introduces the effect of forward

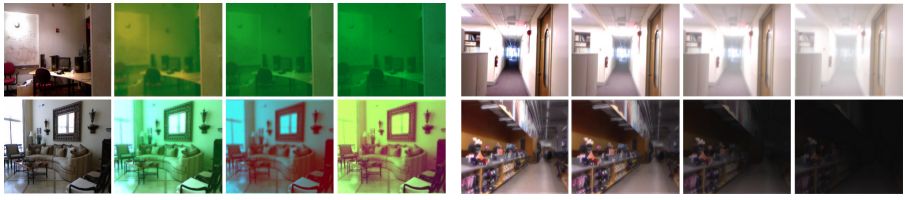


Fig. 1: Examples of degradations that can be simulated using our approach on NYU Depth v2 dataset. Left: First row - underwater degradation (with increased water attenuation); Second row - diverse attenuation-lighting-scattering configurations. Right: first row - scatter-enriched fog, second row - smoke-like degraded environments.

scattering by replacing the known radiance (clear image) with the sum of two contributions: one due to direct or straight-path scene radiance $J'_c(\mathbf{x}) = (1 - k_c(\mathbf{x}))J_c(\mathbf{x})$ and another due to scattered light from scene radiance $J_c^{sct}(\mathbf{x})$, where $k_c(\mathbf{x}) = \exp^{-\alpha_c z(\mathbf{x})}$ modulates the depth-dependent likelihood that photons from the scene follow a straight-line path. α_c parameterizes scattering media (in an analogous fashion to a "particle density"); we assume that all unscattered photons follow straight-line paths which are perpendicular to the degraded image plane (as is implicitly assumed in Equation 1). For forward scattering radiance, we follow an isotropic, non-polarized, Lambertian-inspired diffusion & reflectance [40] light transmission model. In addition, we see scattering interactions as occurring at the same depth as the radiance source. In order to model how scattered photons travel from a single radiance source, we use a diffusive (bi-variate isotropic Gaussian) approximation to compute the likelihood that a scattered photon departing from the radiance source (clear image pixel coordinates) $\mathbf{x}' = (x'_1, x'_2)$ with depth $z(\mathbf{x}')$ arrives at pixel coordinates $\mathbf{x} = (x_1, x_2)$ of the degraded image plane:

$$G_c(\mathbf{x}, \mathbf{x}') = \frac{1}{2\pi(\gamma_c z(\mathbf{x}'))} \exp\left(-\frac{(x'_1 - x_1)^2 + (x'_2 - x_2)^2}{2(\gamma_c z(\mathbf{x}'))^2}\right), \tag{3}$$

where γ_c modulates the scattering relationship with distance for channel c (acting as a proxy for scattering trajectory variance). Eq. 3 can be interpreted as the two dimensional distribution of the forward-scattered arriving intensity over the degraded image plane from a single radiance source; the larger the "on-plane" distance separating source pixel $\mathbf{x}'$ from the recovery pixel $\mathbf{x}$ is, the lower this likelihood becomes (see [41] for a more elaborate version approximating the Lorentz-Mie scattering phase function distribution on individual particles). Furthermore, we view each clear image pixel $\mathbf{x}' \in S$ as an individual radiance source, being each one subject to scattering dynamics independently (this holds if wave-like and quantum interactions are excluded). Therefore, for a given degraded image pixel $\mathbf{x}$ and channel c, integrating the radiance contributions onto $\mathbf{x}$ (e.i. the likelihood of a photon being scattered, multiplied by the likelihood of a scattered photon arriving at $\mathbf{x}$, multiplied by the source radiance) from each source on S, provides an expectation of the recovered scattered signal:

$$J_c^{sct}(\mathbf{x}) = \int_{\mathbf{x}' \in S} k_c(\mathbf{x}')J_c(\mathbf{x}')G_c(\mathbf{x}, \mathbf{x}')d\mathbf{x}'. \tag{4}$$

This way, the signal collected at degraded image pixel **x** after considering signal attenuation, scattering effects and ambiance lighting is approximated by

$$I_c^{sct}(\mathbf{x}) = (J_c^{sct}(\mathbf{x}) + (1 - k_c(\mathbf{x}))J_c(\mathbf{x}))t_c(\mathbf{x}) + (1 - t_c(\mathbf{x}))A_c. \quad (5)$$

Finally, we increase the turbid visual appearance of the degraded image by weighting I_c^{sct} with a Gaussian-smoothed salt & pepper-based noise image SP_c, simulating the random presence of larger particles such as colored dissolved matter:

$$I_c(\mathbf{x}) = uI_c^{sct}(\mathbf{x}) + (1 - u)SP_{c\{sp_{col}, pr_c, \sigma_c\}}, \quad (6)$$

where $u \in [0, 1]$ is an image weighting parameter, sp_{col} is a base particle color, pr_c is the probability of adding a particle on SP's channel c, and σ_c is the deviation of the Gaussian blur applied to SP's channel c.

4 Proposed Method

4.1 Network Architecture

The complete network architecture is depicted in Fig. 2. We have used three parallel *encoder-decoder* models, corresponding to three branches of the global models: *i)* simple degraded image formation model, using estimated depth image (see Equation (1)) *ii)* model for residual learning and *iii)* direct prediction of degraded image. For all these three networks, the input R,G,B image is encoded into feature vectors by using a DenseNet-169 [42] network which is pretrained on ImageNet [43]. Then this vector is then fed into a successive series of upsampling layers in order to construct the final depth map at half of the input resolution of the input image. These upsampling layers and their associated skip-connections form our *decoder*. The proposed decoder model is simple and straightforward [44]. Further architectural details are provided in the supplementary materials. The complete network architecture is depicted in Fig. 2. We have used three independent *encoder-decoder* models in this network. The original clean image is taken as the input in all of these three *encoder-decoder* models. The first *encoder-decoder* is used to predict the gray level depth image (I^{Depth}). The second *encoder-decoder* network is used to predict the RGB *residual* image ($I^{Residue}$) and the third *encoder-decoder* network is used to directly predict the *simulated underwater image* ($I^{Simulated}_{Predicted}$) from the original image. By using the estimated depth image (I^{Depth}), user given *atmospheric light* (A_c) and *attenuation coefficients* (β_c, which is further used to calculate t_c; see Equation (2)), we compute the initial model-based degraded image ($I^{Degraded}_{Initial}$) using Equations (1) and 2. Now, by using $I^{Degraded}_{Initial}$ and $I^{Residue}$, we compute the *estimated simulated image* ($\hat{I}^{Predicted}_{Simulated}$). We apply the third *encoder-decoder* network also to directly estimate the *simulated image* ($I^{Direct}_{Predicted}$) from the input of *RGB* image ($I^{Original}$) only. To sum up, this architecture is based on 3 individual blocks of *encoder-decoder* networks. The objective of the first network is to estimate the depth image which is further used to obtain an initial degraded image by applying the physics induced image formation model (see Equation (1)). The second branch learns the residual by capturing everything that is not modeled by Equation (1) in a data driven manner. The third branch directly predicts the ground truth image.

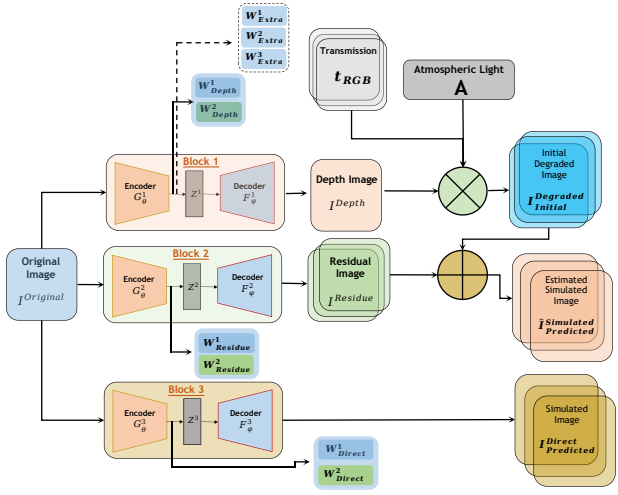


Fig. 2: The proposed network architecture

4.2 Learning Loss Function

Any standard loss function for predicted image regression problems considers a discrepancy measure between the ground truth image and the predicted image. However, different considerations regarding the loss function can have very significant effects on the training speed and the overall performance of image estimation. Here, we have categorically analyzed the influence or significance of each of the three branches of our architecture on the overall performance of the network.

4.2.1 Technique 1: As a first approach, we use the initial two *encoder-decoder* blocks (omitting the third or bottom most *encoder-decoder* block) of the network (see Fig. 2). Our objective here is firstly to define a loss function that balances between the predicted "depth image" (I_{depth}) by minimizing the difference of depth values while also penalizing distortions of high frequency details in the depth image domain (these details typically correspond to object boundaries in the scene). Secondly, we also minimize the difference between *predicted simulated image* $(I^{Simulated}_{Predicted})$ and the manually calculated ground truth version of the same image. Thirdly, based on the predicted *depth image* (I_{depth}), user given *attenuation coefficients* (β_c), which is further used to calculate the "transmission matrix" (t_c) and *atmospheric light* (A), we first generate the "initial degraded image" $(I^{Degraded}_{Initial})$, which is then combined with the estimated "residual" image i.e. $I^{Residue}$ to generate the "estimated simulated image" $\hat{I}^{Predicted}_{Simulated}$ (see Equation (11)). Hence, we need to take into account the correct estimation of I_{depth}, followed by the correct estimation of $I^{Degraded}_{Initial}$ which is the main contribution in the formation of $\hat{I}^{Predicted}_{Simulated}$. It is also equally important to properly estimate $I^{Residue}$ as it will represent the missing quantity between the targeted estimated image $\hat{I}^{Predicted}_{Simulated}$ and $I^{Degraded}_{Initial}$. That is why we add $I^{Degraded}_{Initial}$ and $I^{Residue}$ to obtain the $\hat{I}^{Predicted}_{Simulated}$.

To train using this configuration, we first define two separate loss functions i.e. L_d, L_p and compute the final loss $L_{total} = L_d + L_p$. L_d represents the loss corresponding to the depth image (I_{depth}) reconstruction, and L_p represents the loss corresponds to the "estimated simulated image" ($\hat{I}^{Predicted}_{Simulated}$) reconstruction. L_d can be defined as follows:

$$L_d(y_p, \hat{y}_p) = \lambda_1^y L_{depth}(y_p, \hat{y}_p) + \lambda_2^y L_{SSIM}(y_p, \hat{y}_p) \tag{7}$$

The depth term L_{depth} is the point-wise $L1$ loss which is defined on the depth values :

$$L_{depth}(y_p, \hat{y}_p) = \frac{1}{n}\sum_{p}^{n}|y_p - \hat{y}_p| \tag{8}$$

$$L_{SSIM}(y_p, \hat{y}_p) = \frac{1 - SSIM(y, \hat{y})}{2} \tag{9}$$

The second term L_{SSIM} uses the Structural Similarity (SSIM) [45], a common metric for image reconstruction tasks and it has been shown to be a good loss term for depth estimation by using CNNs [46]. As SSIM ≤ 1, we define a loss L_{SSIM} as in Equation (9). Please note that, in Equation (7), we only have defined λ_1^y and λ_2^y as two weighting parameters which we have empirically set to $\lambda_1^y = \lambda_2^y = 0.1$. The inherited problem with such loss terms is that they tend to be larger when the ground-truth depth values are bigger. In order to resolve this issue, the reciprocal depth values are considered [47], where the original depth map y_{orig} is replaced by the target depth map $y = m/y_{orig}$; where m is the maximum depth in the scene (e.g. $m = 10$ meters for the NYU Depth v2 dataset). Our approach considers transforming the depth values and computing the loss in the log space [48], [49]. The second loss L_p is defined as:

$$L_p(q_p, \hat{q}_p) = \lambda_1^q \left[\frac{1}{n}\sum_{p}^{n}|q_p - \hat{q}_p|\right] + \lambda_2^q \left[\frac{1 - SSIM(m_p, \hat{q}_p)}{2}\right] \tag{10}$$

where q_p and $\hat{q}_p$ represents true and predicted "estimated simulated image" ($\hat{I}^{Predicted}_{Simulated}$).

$$\begin{aligned} &I^{Degraded}_{Initial}(\mathbf{x}) = t_c(\mathbf{x}) I_{original}(\mathbf{x}) + (1 - t_c(\mathbf{x})).A_c \\ &t_c(\mathbf{x}) = \exp^{-\beta_c I_{Depth}(\mathbf{x})}; \qquad \hat{I}^{Simulated}_{Predicted}(\mathbf{x}) = I^{Degraded}_{Initial}(\mathbf{x}) + I^{Residue} \end{aligned} \tag{11}$$

Finally, the total loss is calculated as follows : $L_{total} = L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p)$. Furthermore, for this technique and the next ones, we also propose following two more variants to compute the loss function.

4.2.1.1 Variant 1 : Instead of fixing the values of λ_1^q and λ_2^q as 0.1, we compute them automatically by using the same network, shown in Fig. 2. After obtaining the pre-trained features from the last encoder block of *DenseNet*, those are passed through two blocks of *Fully Connected (FC) Layers*. These features are then flattened and reduced in dimension by passing through several linear layers with ReLU activations. Finally these features are passed through *sigmoid* layer to obtain two weights (see Block-1 and Block-2 in Fig. 2. Further architectural details are mentioned in supplementary

material)[3]. By using these automatic weight values, the L_d and L_p is calculated as:

$$\begin{aligned} L_d(y_p, \hat{y}_p) &= w^1_{Depth} L_{depth}(y_p, \hat{y}_p) + (1 - w^1_{Depth}) L_{SSIM}(y_p, \hat{y}_p) \\ L_p(q_p, \hat{q}_p) &= w^1_{Residue} \left[\frac{1}{n}\sum_p^n |q_p - \hat{q}_p|\right] + (1 - w^1_{Residue}) \left[\frac{1 - SSIM(q_p, \hat{q}_p)}{2}\right] \\ L_{total} &= L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) \end{aligned} \tag{12}$$

4.2.1.2 Variant 2 : In addition with the weighted computation of $L_d(y_p, \hat{y}_p)$ and $L_p(q_p, \hat{q}_p)$ (according to Equation (12)), we also compute the total loss in the following manner.

$$L_{total} = w^2_{Depth} L_d(y_p, \hat{y}_p) + (1 - w^2_{Depth}) L_p(q_p, \hat{q}_p) \tag{13}$$

4.2.2 Technique 2: As a second approach, a loss term corresponding to the "initial degraded" image (computed by using Equation (1)) is added, compared to the total loss of Equation (12). Along with the loss of depth image (I_{Depth}) and "estimated simulated image" ($\hat{I}^{Predicted}_{Simulated}$), we compute the loss (denoted as L_t) on $I^{Degraded}_{Inital}$ as:

$$L_t(h_p, \hat{h}_p) = \lambda^h_1 \left[\frac{1}{n}\sum_p^n |h_p - \hat{h}_p|\right] + \lambda^h_2 \left[\frac{1 - SSIM(h_p, \hat{h}_p)}{2}\right] \tag{14}$$

where h_p and $\hat{h}_p$ represents the true and predicted "initial degraded" image ($I^{Degraded}_{Initial}$), and λ^h_1, λ^h_2 are set to 0.1. Finally, the total loss is computed by:

$$L_{total} = L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) + L_t(h_p, \hat{h}_p) \tag{15}$$

As the *variant 1* under this category, here also we compute the weighted (automatic) version of $L_t(h_p, \hat{h}_p)$ in addition to the weighted version of $L_d(y_p, \hat{y}_p)$ and $L_p(q_p, \hat{q}_p)$ (as in Equation (12),) as:

$$\begin{aligned} L_t(h_p, \hat{h}_p) &= w^2_{Depth} \left[\frac{1}{n}\sum_p^n |h_p - \hat{h}_p|\right] + (1 - w^2_{Depth}) \left[\frac{1 - SSIM(h_p, \hat{h}_p)}{2}\right] \\ L_{total} &= L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) + L_t(h_p, \hat{h}_p) \end{aligned} \tag{16}$$

As the *variant 2* under this category, the total loss is computed as (see supplementary material for more details about the architecture).

$$\begin{aligned} L_{total} = w^1_{Extra} \times L_d(y_p, \hat{y}_p) + w^2_{Extra} \times L_p(q_p, \hat{q}_p) \\ + [1 - (w^1_{Extra} + w^2_{Extra})] \times L_t(h_p, \hat{h}_p) \end{aligned} \tag{17}$$

4.2.3 Technique 3: As the third modification, here we introduce an additional *encoder-decoder* block to directly estimate the simulated image ($I^{Direct}_{Predicted}$) from the clean *RGB* image. Hence, we compute a dedicated loss (L_g) for ($I^{Direct}_{Predicted}$) image only:

$$L_g(s_p, \hat{s}_p) = \lambda^s_1 \left[\frac{1}{n}\sum_p^n |s_p - \hat{s}_p|\right] + \lambda^s_2 \left[\frac{1 - SSIM(s_p, \hat{s}_p)}{2}\right] \tag{18}$$

[3] For this architectural configuration, we do not create the second branch from **Block 1**, corresponding to W^1_{Extra}, W^2_{Extra}, W^3_{Extra} as we don't need these weights

where s_p and $\hat{s}_p$ represents the true and predicted "directly simulated" image ($I_{Initial}^{Degraded}$) by the network and the value of λ_1^s and λ_2^s are set to 0.1. Finally, the total loss is:

$$L_{total} = L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) + L_t(h_p, \hat{h}_p) + L_g(s_p, \hat{s}_p) \tag{19}$$

As the *variant 1* under this category, here also we compute the weighted (automatic) version of $L_g(s_p, \hat{s}_p)$ in addition to the weighted version of $L_d(y_p, \hat{y}_p)$, $L_p(q_p, \hat{q}_p)$ and $L_t(h_p, \hat{h}_p)$ (in the same manner as in Equation (16)) in the following manner.

$$\begin{aligned} L_g(s_p, \hat{s}_p) &= w_{Direct}^1 \left[\frac{1}{n}\sum_p^n |s_p - \hat{s}_p|\right] + (1 - w_{Direct}^1) \left[\frac{1-SSIM(s_p,\hat{s}_p)}{2}\right] \\ L_{total} &= L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) + L_t(h_p, \hat{h}_p) + L_g(s_p, \hat{s}_p) \end{aligned} \tag{20}$$

As the *variant 2* under this category, the total loss is computed in the following manner (see supplementary material for more details about the architecture).

$$\begin{aligned} L_{total} = w_{Extra}^1 \times L_d(y_p, \hat{y}_p) + w_{Extra}^2 \times L_p(q_p, \hat{q}_p) + w_{Extra}^3 \\ \times L_t(h_p, \hat{h}_p) + [1 - (w_{Extra}^1 + w_{Extra}^2 + w_{Extra}^3)] \times L_t(h_p, \hat{h}_p) \end{aligned} \tag{21}$$

5 Experimental Results

5.1 Datasets

We have used following two well known datasets which provide *RGB-D* images.

5.1.1 NYU Depth v2: This dataset provides images and depth maps for different indoor scenes, captured at the resolution of 640×480 [50]. For this work, we have used a subset of $50k$ images which we obtained from [51]. The depth map has an upper bound of 10 meters. Like in [51], our method also produces the predictions at the half of the input resolution i.e. at 320×240 and here also we do not crop any of the input image-depth map pairs even though they contain missing pixels, due to the preprocessing for distortion correction. We have used 96% i.e. 48650 images for training and the remaining 2030 images are used for testing purposes.

5.1.2 Make3D: This dataset contains 534 RGB-depth pairs, split into 400 pairs for training and 134 for testing. The RGB images are provided at high resolution while the available depth maps are comparatively at very low resolution. Therefore, the data is resized into 460×345 as proposed in [52] [53]. We used the same data reading and processing protocol as in [54] and the results are evaluated by using depth cap of $0 - 80$.

5.2 Evaluation

We use standard metrics [48] to quantitatively compare our method against state-of-the-art techniques. These error metrics are defined as:

i. Average relative error (rel): $\frac{1}{n}\sum_p^n \frac{|y_p - \hat{y}_p|}{y}$

ii. Root mean square error (rms): $\sqrt{\frac{1}{n}\sum_p^n (y_p - \hat{y}_p)^2}$

iii. Average ($\log_{10}$) error: $\frac{1}{n}\sum_p^n |\log_{10}(y_p) - \log_{10}(\hat{y}_p)|$

iv. Threshold accuracy (δ_i): % of y_p s.t. $max(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}) = \delta < Thersh$ for $Thresh = 1.25, 1.25^2, 1.25^3$;

Where y_p is a pixel in depth image y, $\hat{y}_p$ is a pixel in the predicted depth image $\hat{y}$ and n is the total number of pixels for each depth image.

In Tables 1, we report the accuracy of the NYU-V2 dataset. The performance of 9 variants of the proposed methods are compared with 6 other relevant techniques. The *variant-1* for each technique performs slightly better than the core technique (i.e. *Technique-1 Variant-1* has performed better than *Technique-1* itself). Whereas *variant-2* outperforms the *variant-1* by a large margin. These results signifies that performing the weighted combination of the contribution from different entities in the total loss (see Equation (13)) strongly improves the results compared to considering equal and full contribution of each term. Now, if we compare *Technique-1*, *Technique-2* and their

Table 1: Comparisons of different methods on the NYU Depth v2 dataset

Method	$\boldsymbol{\delta_1}\uparrow$	$\boldsymbol{\delta_2}\uparrow$	$\boldsymbol{\delta_3}\uparrow$	$\boldsymbol{rel}\downarrow$	$\boldsymbol{rms}\downarrow$	$\boldsymbol{\log_{10}}\downarrow$
Proposed: Technique-1	0.505	0.747	0.890	0.614	0.176	0.150
Proposed: Technique-1 Variant-1	0.512	0.750	0.892	0.633	0.174	0.150
Proposed: Technique-1 Variant-2	**0.769**	**0.947**	**0.986**	**0.165**	**0.109**	**0.065**
Proposed: Technique-2	0.507	0.748	0.888	0.610	0.172	0.150
Proposed: Technique-2 Variant-1	0.509	0.736	0.885	0.622	0.173	0.150
Proposed: Technique-2 Variant-2	**0.779**	**0.945**	**0.984**	**0.163**	**0.105**	**0.065**
Proposed: Technique-3	0.510	0.743	0.883	0.601	0.173	0.150
Proposed: Technique-3 Variant-1	0.514	0.748	0.895	0.627	0.173	0.149
Proposed: Technique-3 Variant-2	***0.792***	***0.952***	***0.987***	***0.152***	***0.100***	***0.060***
Encoder-Decoder Model [51]	0.269	0.556	0.776	1.121	0.2345	0.231
Pix2Pix [25]	0.743	0.900	0.957	0.204	0.069	0.080
CycleGAN [30]	0.227	0.418	0.572	1.242	0.303	0.315
UNIT [28]	0.220	0.402	0.559	1.194	0.307	0.334
MUNIT [32]	0.233	0.364	0.482	1.394	0.341	0.349
DRIT [33]	0.246	0.451	0.606	1.223	0.311	0.298

variants, there is only minute difference in results within these approaches. Compared to *Technique-1*, in *Technique-2* the additional loss (i.e. $L_t(h_p, \hat{h}_p)$) of "initial degraded" (i.e. $I_{Initial}^{Degraded}$) image is added. As we have already used the depth image loss (L_d) in *Technique-1* and *Technique-2*, adding the L_t loss for $I_{Initial}^{Degraded}$ image, which is computed by using the depth image (i.e. I_{Depth}) and other user given constants i.e. $I_{Original}$, β_c and A_c (see Equation (11)) does not make much difference in terms of loss level contribution. Furthermore, in *Technique-3* we add the additional loss L_g, corresponding to $I_{Predicted}^{Direct}$ image which also inherently adds computational burden, related to the third block of *encoder-decoder* network (see Fig. 2). But adding this extra loss of L_g could only slightly improve the performance. Hence, we should find a

Table 2: Comparisons of different methods on the Make-3D dataset

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	$rel \downarrow$	$rms \downarrow$	$\log_{10} \downarrow$
Proposed: Technique-1	0.359	0.603	0.757	1.613	0.284	0.226
Proposed: Technique-1 Variant-1	0.346	0.608	0.767	1.55	0.267	0.223
Proposed: Technique-1 Variant-2	0.355	0.617	0.772	**1.527**	**0.262**	**0.220**
Proposed: Technique-2	**0.383**	**0.627**	0.771	1.760	0.292	0.223
Proposed: Technique-2 Variant-1	0.366	0.615	0.769	1.59	0.270	0.221
Proposed: Technique-2 Variant-2	0.362	0.625	**0.773**	1.60	0.269	**0.220**
Proposed: Technique-3	**0.379**	0.621	0.771	1.648	0.281	**0.220**
Proposed: Technique-3 Variant-1	0.330	0.604	0.766	1.905	0.271	0.232
Proposed: Technique-3 Variant-2	0.365	**0.626**	**0.778**	1.589	**0.263**	**0.218**
Encoder-Decoder Model [51]	0.331	0.619	***0.808***	**1.485**	***0.242***	***0.211***
Pix2Pix [25]	0.239	0.458	0.638	***1.367***	0.301	0.264
CycleGAN	***0.654***	***0.698***	0.741	1.865	0.760	0.567

good trade-off between leveraging slight improvement in accuracy compared to the additional computational burden. The values of δ_1, δ_2 and δ_3 (which count the number of pixels which are similar to each other between the ground truth and predicted image with respect to three different threshold values) shows that the *variant-2* of all the three techniques have shown strong improvement in accuracy compared the core technique and *variant-1*. However, the error metrics i.e. *rel*, *rms* and $\log_{10}$ (which calculates the pixel level errors between the ground truth and predicted image) shows that *variant-2* of all the three techniques have shown strong improvement in accuracies.

The proposed techniques are compared with several other relevant image-to-image translation approaches. As the first such technique, we use only a single *encoder-decoder* model (see third block in Fig.2) similar to the technique in [51] to directly generate the degraded image from a *RGB* image. Except Pix2Pix [25], all other comparable techniques do not perform well enough. Although Pix2Pix [25] could perform better than other state-of-the-art techniques, this image-to-image translation approach is simply based on classical *conditional-adversarial loss* and *L1* loss which does not incorporate any physical image degradation model/equation like our proposed technique. The top 3 results for each metric is noted in bold whereas the best result is mentioned in bold-italic style in Table.1 and Table.2.

A few examples of the proposed *Technique 1* is shown in Fig. 3 where we can observe that to a good extent the proposed technique is able to generate "initial degraded image" ($I_{Initial}^{Degraded}$) (see Fig. 3d) with respect to the ground truth, as shown in Fig. 3b. There are substantial differences between the "simulated underwater image" (I_c^{sct}) and $I_{Initial}^{Degraded}$ image (see Fig. 3c and Fig. 3b). The "haze image" explains some of the physical degradation in an interpretable way, but needs to be complemented by the data-driven residual. The proposed model can successfully capture this substantial difference i.e. the residual image is shown in Fig. 3e. By using these $I_{Initial}^{Degraded}$ and $I^{Residue}$, we can successfully predict the "Simulated Underwater Image" ($\hat{I}_{Predicted}^{Simulated}$), shown in Fig. 3f. Still, we are not always able to correctly reconstruct the colors perfectly but we can simulate the strong underwater blurry degradation effects.

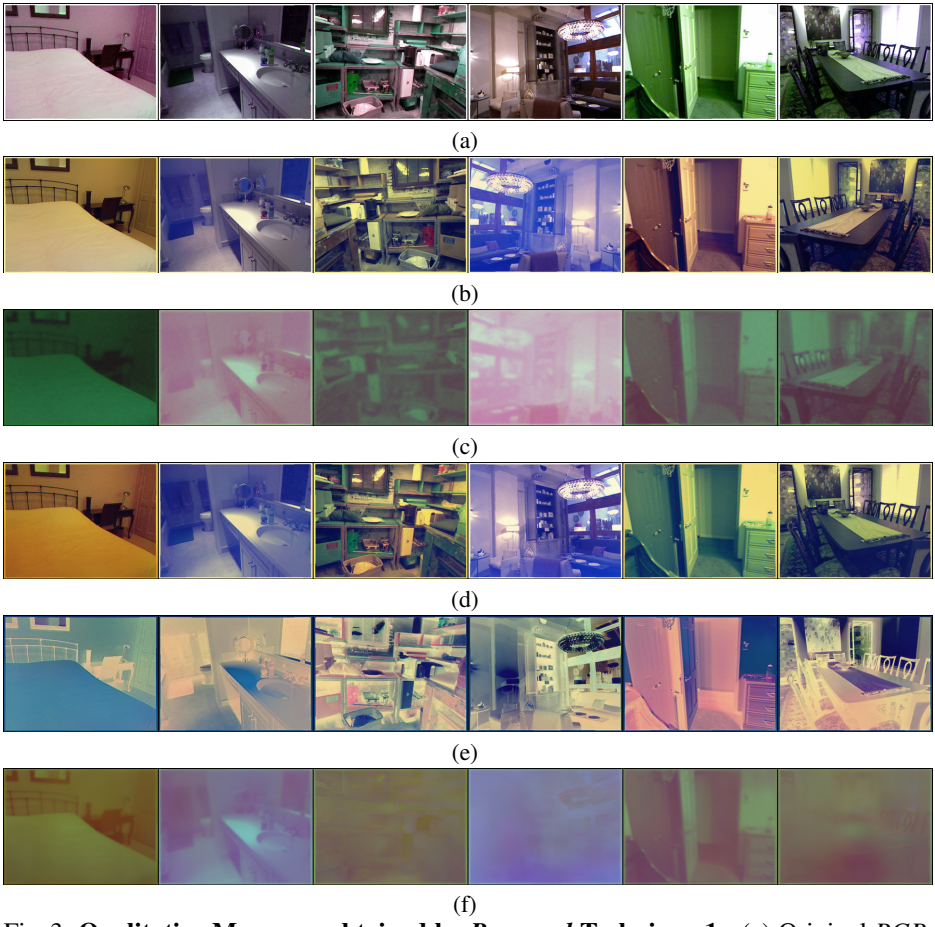


Fig. 3: **Qualitative Measures obtained by *Proposed* Technique 1 :** (a) Original *RGB* images (i.e. $J_c(\mathbf{x})$ in Equation (1)) (b) Ground Truth of "Initial Degraded Image" (i.e. $I_c(\mathbf{x})$ in Equation (1)) (c) Ground Truth of "Simulated Underwater Image" (i.e. $I_c^{sct}(\mathbf{x})$ in Equation (5)) (d) Predicted "Initial Degraded Image" (i.e. $I_{Initial}^{Degraded}$ in Fig. 2) (e) Predicted "Residual Image" (i.e. $I^{Residue}$ in Fig. 2) (f) Predicted "Simulated Underwater Image" (i.e. $\hat{I}_{Predicted}^{Simulated}$ in Fig. 2).

It is difficult to draw any substantial conclusion from the results of Make-3D dataset (as is very small and the depth image resolution is quite low, compared to *RGB* image, making it necessary to interpolate the depth image which highly degrades its quality), shown in Table 2. The same phenomenon is also visible that *variant-2* has performed either better or very similar to it's counterparts i.e. the core technique and *variant-1* for all of the three proposed techniques. By observing the weaker performance of other state-of-the-art methods from Table. 1, here we only have tested the simple *encoder-decoder* model and most relevant as well as well known *Pix2Pix*, *CycleGAN* networks. The performance of these techniques are either close (e.g. *encoder-decoder* network) or fall behind (e.g. *Pix2Pix* model) the proposed techniques. Moreover, for certain metric

(i.e. δ_1 and δ_2), *CycleGAN* has outperformed others. Most importantly, as mentioned before that none of these technique incorporate any physical image degradation model/equation and are as interpretable as the proposed methods.

6 Discussion

An interesting byproduct of our approach is that our trained model enables us to solve the inverse problem of underwater image restoration. Indeed, suppose we want to obtain a clean image J_c from a degraded underwater image y. A natural way of carrying out this task is to minimize a mean squared error between the output of a known forward physical model f and the real underwater image:

$$\arg\min_{J_c,\theta} ||f(J_c, \theta) - y||^2$$

w.r.t. the input parameters of the model θ, in our case veiling light, attenuation coefficients, depth image, and also the clean image J_c itself; depth image could be estimated from degraded underwater image y using a learned model. The resulting optimization problem requires computing the derivatives of f: $\frac{\partial f}{\partial \theta}$ and $\frac{\partial f}{\partial J_c}$. Analytic derivatives are typically cumbersome to obtain for intricate physical models; hence leveraging automatic differentiation tools (e.g. Pytorch, Tensorflow etc.) is necessary. However, this requires f to be perfectly known, and implemented in a modern automatic differentiation package, [55]. Here, the complex image formation model needs to be both known and differentiable. In real scenarios, a generative physical model is often unknown or badly known. Even when it is known, it is often implemented in languages that do not support automatic differentiation (e.g. C++, Fortran). With our approach, we obtain a deep learning-based emulator [55], in Pytorch: obtaining the derivatives of this model w.r.t to any of its inputs is straightforward thanks to it's automatic differentiation property. Moreover, the emulator does not require the knowledge of the underlying governing equations, its only task is to reproduce the desired outputs, even in a black-box fashion. Therefore, implementing algorithms to solve inverse problems which require to optimize over the model's input or parameters is easy with our trained model, and will be a basis for our future work.

7 Conclusion

In this paper, we have proposed a physics-informed and data-driven deep learning architecture to simulate the effect of underwater image degradation. We proposed a complex image formation model to create a simulated dataset from any RGB-depth available dataset. We proposed to inform our network with a simple haze image formation model that is able to account for simple image degradations, provided a good estimate of the depth image can be obtained. This image, as well as a residual image that captures the missing physics directly from data are obtained via DenseNet encoding-decoding blocks. Different losses are designed in order to estimate each component as well as their weighting parameters; which are obtained automatically. We obtain an emulator

of this physical phenomenon that paves the way to obtaining differentiable and efficient emulators of complex physical models in other scenarios. We have shown on two datasets that our approach outperforms classical physics-ignorant deep learning models suited for image to image translation tasks. Future work will revolve around exploiting the interpretability and differentiability of the model to solve the inverse problem of image restoration.

References

1. C. D. Mobley. Light and Water: Radiative transfer in natural waters (vol. 592). *Ligth and Water : Radiative Transfer in Natural Waters*, (January 1994):554, 2004.
2. Derya Akkaynak, Tali Treibitz, Tom Shlesinger, Raz Tamir, Yossi Loya, and David Iluz. What is the space of attenuation coefficients in underwater computer vision? *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:568–577, 2017.
3. N.G. Jerlov. Marine optics. *Marine Optics*, 1(5):455–456, 1977.
4. Armin Schwartzman, Marina Alterman, Rotem Zamir, and Yoav Y. Schechner. Turbulence-induced 2D correlated image distortion. In *2017 IEEE International Conference on Computational Photography, ICCP 2017 - Proceedings*, 2017.
5. Dana Berman, Tali Treibitz, and Shai Avidan. Non-Local Image Dehazing. In *CVPR*, volume 2016-Decem, pages 1674–1682, 2016.
6. Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2341–2353, 2011.
7. Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *CVPR*, 33(12):2341–2353, 2009.
8. Raanan Fattal. Dehazing using color-lines. *ACM Transactions on Graphics*, 34(1), 2014.
9. Ko Nishino, Louis Kratz, and Stephen Lombardi. Bayesian defogging. *International Journal of Computer Vision*, 98(3), 2012.
10. Robby T. Tan. Visibility in bad weather from a single image. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.
11. Nicholas Carlevaris-Bianco, Anush Mohan, and Ryan M. Eustice. Initial results in underwater single image dehazing. In *MTS/IEEE Seattle, OCEANS 2010*, 2010.
12. John Y. Chiang and Ying Ching Chen. Underwater image enhancement by wavelength compensation and dehazing. *IEEE Transactions on Image Processing*, 21(4):1756–1769, 2012.
13. P. Drews-Jr, E. Do Nascimento, F. Moraes, S. Botelho, and M. Campos. Transmission estimation in underwater single images. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.
14. Adrian Galdran, David Pardo, Artzai Picón, and Aitor Alvarez-Gila. Automatic Red-Channel underwater image restoration. *Journal of Visual Communication and Image Representation*, 26, 2015.
15. Huimin Lu, Yujie Li, Lifeng Zhang, and Seiichi Serikawa. Contrast enhancement for images in turbid water. *Journal of the Optical Society of America A*, 32(5), 2015.
16. Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters - Improve semantic segmentation by global convolutional network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1743–1751, 2017.
17. Cosmin Ancuti, Codruta Orniana Ancuti, Tom Haber, and Philippe Bekaert. Enhancing underwater images and videos by fusion. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012.
18. Huimin Lu, Yujie Li, and Seiichi Serikawa. Underwater image enhancement using guided trigonometric bilateral filter and fast automatic color correction. In *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, 2013.
19. Cosmin Ancuti, Codruta O. Ancuti, Christophe De Vleeschouwer, Rafael Garcia, and Alan C. Bovik. Multi-scale underwater descattering. In *Proceedings - International Conference on Pattern Recognition*, volume 0, 2016.

20. Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. DehazeNet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, 2016.
21. Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. AOD-Net: All-in-One Dehazing Network. *Iccv*, pages 4770–4778, 2017.
22. Wei Ting Chen, Jian Jiun Ding, and Sy Yen Kuo. PMS-Net: Robust haze removal based on patch map for single images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:11673–11681, 2019.
23. Wenqi Ren, Si Liu, Hua Zhang, Jinshan Pan, Xiaochun Cao, and Ming Hsuan Yang. Single image dehazing via multi-scale convolutional neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9906 LNCS:154–169, 2016.
24. Young Sik Shin, Younggun Cho, Gaurav Pandey, and Ayoung Kim. Estimation of ambient light and transmission map with common convolutional architecture. *OCEANS 2016 MTS/IEEE Monterey, OCE 2016*, 2016.
25. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. 2016.
26. Yunjey Choi, Minje Choi, Munyoung Kim, Jung Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.
27. Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–14, 2017.
28. Ming Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):701–709, 2017.
29. Zili Yi, Hao Zhang, Ping Tan, Minglun Gong, and C V Oct. DualGAN : Unsupervised Dual Learning for Image-to-Image Translation.
30. Jun Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:2242–2251, 2017.
31. Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *34th International Conference on Machine Learning, ICML 2017*, 4:2941–2949, 2017.
32. Xun Huang, Ming Yu Liu, Serge Belongie, and Jan Kautz. Multimodal Unsupervised Image-to-Image Translation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11207 LNCS:179–196, 2018.
33. Hsin Ying Lee, Hung Yu Tseng, Jia Bin Huang, Maneesh Singh, and Ming Hsuan Yang. Diverse Image-to-Image Translation via Disentangled Representations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11205 LNCS:36–52, 2018.
34. Jun Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. *Advances in Neural Information Processing Systems*, 2017-Decem(1):466–477, 2017.
35. Yoav Y. Schechner and Nir Karpel. Recovery of underwater visibility and structure by polarization analysis. *IEEE Journal of Oceanic Engineering*, 30(3):570–587, 2005.
36. Curtis Mobley and Sequoia Scientific. Light and Water : Radiative Transfer in Natural. *Ligth and Water : Radiative Transfer in Natural Waters*, (January 1994), 2016.
37. N. G. (Nils Gunnar) Jerlov. Marine optics. page 231, 1976.

38. Qiang Fu and Wenbo Sun. Mie theory for light scattering by a spherical particle in an absorbing medium. *Applied Optics*, 40(9):1354–1361, 2001.
39. Annick Bricaud, André Morel, Marcel Babin, Karima Allali, and Hervé Claustre. Variations of light absorption by suspended particles with chlorophyll a concentration in oceanic (case 1) waters: Analysis and implications for bio-optical models. *Journal of Geophysical Research: Oceans*, 103(C13):31033–31044, 1998.
40. Shree K Nayar, Katsushi Ikeuchi, and Takeo Kanade. Surface reflection: Physical and geometrical perspectives. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 13(7), 1991.
41. Edouard Berrocal, David L Sedarsky, Megan E Paciaroni, Igor V Meglinski, and Mark A Linne. Laser light scattering in turbid media part i: Experimental and simulated results for the spatial intensity distribution. *Optics express*, 15(17):10649–10665, 2007.
42. Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 2261–2269, 2017.
43. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. 2010.
44. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, 2015.
45. Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.
46. Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, 2017.
47. Po Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia Bin Huang. DeepMVS: Learning Multi-view Stereopsis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.
48. David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, volume 3, pages 2366–2374, 2014.
49. Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 5622–5631, 2017.
50. Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7576 LNCS(PART 5):746–760, 2012.
51. Ibraheem Alhashim and Peter Wonka. High Quality Monocular Depth Estimation via Transfer Learning. 2018.
52. Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, 2005.
53. Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Learning 3-D scene structure from a single still image. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.
54. Shir Gur and Lior Wolf. Single image depth estimation trained via depth from defocus cues. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, 2019.

55. Marcel Nonnenmacher and David S Greenberg. Deep emulators for differentiation, forecasting, and parametrization in earth science simulators. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002554, 2021.
56. Dana Berman, Tali Treibitz, and Shai Avidan. Diving into haze-lines: Color restoration of underwater images. *British Machine Vision Conference 2017, BMVC 2017*, pages 1–12, 2017.
57. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Supplementary Materials

No Author Given

No Institute Given

1 More Details on Image Formation Model

1.1 Classical Simulation Model

The classical image formation model (developed in [35]) is written in Equation (1). The image intensity (**x**) at each pixel is composed of two components: the attenuated signal and the veiling light.

$$I_c(\mathbf{x}) = t_c(\mathbf{x})J_c(\mathbf{x}) + (1 - t_c(\mathbf{x})).A_c \tag{1}$$

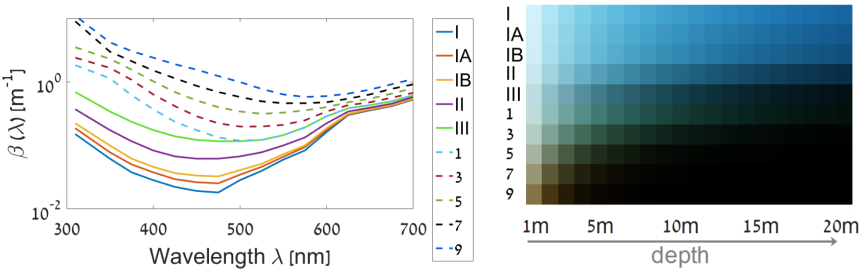
where bold letters denotes vectors, **x** is the pixel coordinate, I_c is the acquired image value in the color channel c, t_c is the transmission of the color channel, and J_c is the object radiance or clean image. The global veiling-light/atmospheric light component A_c is the scene value in the areas with no objects ($t_c = 0, \forall c \in \{R, G, B\}$). The transmission depends on the object's distance $z(\mathbf{x})$ and the attenuation coefficient of the medium for each channel i.e. β_c:

$$t_c(\mathbf{x}) = \exp^{-\beta_c z(\mathbf{x})} \tag{2}$$

In the ocean, the attenuation of red colors can be an order of magnitude larger than the attenuation of blue and green [36]. Hence, contrary to the common assumption in single image dehazing, the transmission $\mathbf{t} = (t_R, t_G, t_B)$ is wavelength dependent.

1.2 Water Attenuation

The attenuation of light in underwater is not constant and varies with the change in geography, seasons and climate. The attenuation coefficient (β) is dependent on wavelength of various water types. For clear open waters, the longest wavelength of visible light is first absorbed, resulting in deep blue colors to the eye. Waters near to the shore contain more suspended particles than the central ocean waters which scatter light and make coastal waters less clear than open waters. Moreover, the absorption of shortest wavelengths is stronger, thus the green wavelength reaches deeper than the other wavelengths. Based on the water clarity, Jerlov [37] proposed a classification scheme for oceanic waters where open ocean waters are classified into class **I**, **IA**, **IB**, **II** and **III**. He also defined the water type **1** through **9** for coastal waters. Type **I** is the clearest and type **III** is the most turbid open ocean water. Similarly, for coastal water, type **1** is the clearest and type **9** is the most turbid. We use three attenuation coefficients: i.e. $\beta_R, \beta_G, \beta_B$ corresponding to RGB channels for our work.



(a) (b)

Fig. 1: (a) The attenuation coefficients (β) of Jerlov water types. The solid lines corresponds to open ocean water types while the dashed lines mark coastal water types. (b) For the case of different water types, the simulation of the appearance of white surface, viewed at depth of $1-20$m. (figures are taken from [56])

The Fig. 1a shows the attenuation coefficient (β) dependency on wavelength of various water types. Whereas, Fig. 1b shows an RGB simulation of the appearance of a perfect white surface viewed at different depths in different water types. A common notion is followed that red colors gets attenuated faster than blue/green colors only in the case of ocean water types. Based on three color channel i.e. R,G,B, we are interested in three corresponding attenuation coefficients: i.e. $\beta_R, \beta_G, \beta_B$ in order to correct/denoise the image. In our work, we use only Jerlov water types to constrain the space of attenuation coefficients in the RGB domain.

2 Network Architecture

Our encoder-decoder network is based on *DenseNet-169* [42] where top most layer related to original ImageNet classification task is removed. Here, we use a pretrained *DenseNet-169* model, which was trained on *ImageNet* dataset [57]. The encoder structure is shown in Table. 1 where the initial image of size $b \times 3 \times 480 \times 640$ (where b is the batch size), is sequentially passed through several layers, which are mentioned in each row of the Table. 1. The size of the image gets gradually decreased by passing through each layer of the network and output size from each layers are mentioned in 2^{nd} column of the Table. 1. For more details about the *DenseNet-169* network, please see [42]. Please note that we have used pretrained model to easily obtain the image features from every *DenseNet-169* network layers.

For the decoder part, we pass the output of *Batch Norm -5* layer (see row 13 of Table. 1) through *ReLU* activation, succeeded by the following convolutional layer (see Table. 2), where $\beta = F$ and $\zeta = F$ to generate a tensor of size $[t1 : b \times F \times m \times n]$; where $m = 15$, $n = 20$ and $F = 1664$ respectively. Now, to concatenate the output tensor (*Trans-2*) from *Transition Layer-2* (see item 9 in Table.1) i.e. the tensor of $[b \times 256 \times 30 \times 40]$ dimension with $t1$, we perform an up-sampling to double the size of $t1$ by using bi-linear interpolation which generates a tensor of size $[b \times F \times (m \times 2) \times (n \times 2)]$. Now the tensor $t1$ and *Trans-2* are concatenated along it's 2^{nd} dimension to generate another tensor of $[b \times 1920 \times (m \times 2) \times (n \times 2)]$ dimension. Then this tensor is

Table 1: DenseNet-169 based encoder model

Sl. No.	Layers	Output Size	DenseNet-169
1	Input Image	$b \times 3 \times 480 \times 640$	$\times$
2	Convolution	$b \times 64 \times 240 \times 320$	No. of filters : 64; Conv: 7×7; Stride: 2×2; Padding: 3×3
3	Batch Norm	$b \times 64 \times 240 \times 320$	$\epsilon = 10^{-5}$; momentum = 0.1
4	ReLU Activation	$b \times 64 \times 240 \times 320$	
5	Max Pooling	$b \times 64 \times 120 \times 160$	Kernel Size: 3×3; Stride: 2×2; Padding: 1×1; Dilation: 1×1
6	Dense Block - 1	$b \times 256 \times 120 \times 160$	$\left[\begin{array}{c}\text{Conv: } 1 \times 1 \\ \text{Conv: } 3 \times 3\end{array}\right] \times 6$
7	Transition Layer - 1	$b \times 128 \times 120 \times 160$	Conv: 1×1
		$b \times 128 \times 60 \times 80$	Average Pool : 2×2; Stride: 2×2
8	Dense Block - 2	$b \times 512 \times 60 \times 80$	$\left[\begin{array}{c}\text{Conv: } 1 \times 1 \\ \text{Conv: } 3 \times 3\end{array}\right] \times 12$
9	Transition Layer - 2	$b \times 256 \times 30 \times 40$	Conv: 1×1
		$b \times 256 \times 30 \times 40$	Average Pool : 2×2; Stride: 2×2
10	Dense Block - 3	$b \times 1280 \times 30 \times 40$	$\left[\begin{array}{c}\text{Conv: } 1 \times 1 \\ \text{Conv: } 3 \times 3\end{array}\right] \times 32$
11	Transition Layer - 3	$b \times 640 \times 15 \times 20$	Conv: 1×1
		$b \times 640 \times 15 \times 20$	Average Pool : 2×2; Stride: 2×2
12	Dense Block - 4	$b \times 1664 \times 15 \times 20$	$\left[\begin{array}{c}\text{Conv: } 1 \times 1 \\ \text{Conv: } 3 \times 3\end{array}\right] \times 6$
13	Batch Norm - 5	$b \times 1664 \times 15 \times 20$	$\epsilon = 10^{-5}$; momentum = 0.1

Table 2: The convolution layer

Convolution	No. of input filters/channels : β; No. of output filters/channels : ζ; Conv kernel : 1×1; Stride: 1×1; Padding: 0

passed through *Convolution-A* layer (see item 1 of Table. 3) to generate a tensor of size $[b \times 832 \times (m \times 2) \times (n \times 2)]$; where β and ζ is taken as 1920 and $\frac{F}{2} = 832$ respectively. Then this generated tensor is sequentially passed through *Leaky ReLU-A*, *Convolution-B* and *Leaky ReLU-B* layers (see items 2, 3 and 4 in Table. 3) to generate the tensor of size $[t2 : b \times 832 \times (m \times 2) \times (n \times 2)]$; where $\beta = \zeta = 832$ are taken for *Convolution-B* layer (see item 3 in Table. 3).

Table 3: The up-sampling block

Sl. No.	Layers	Specifications
1	Convolution-A	No. of input filters/channels : β; No. of output filters/channels : ζ; Conv kernel : 3×3; Stride: 1; Padding: 1
2	Leaky ReLU-A	$\alpha = 0.2$; where α controls the angle of the negative slope
3	Convolution-B	No. of input filters/channels : β; No. of output filters/channels : ζ; Conv kernel : 3×3; Stride: 1; Padding: 1
4	Leaky ReLU-B	$\alpha = 0.2$

Now in the same manner, the tensor $t2$ is interpolated and concatenated with the output tensor (*Trans-1*) from *Transition Layer-1* (see item 7 in Table.1) i.e. the tensor of $[b \times 128 \times 60 \times 80]$ dimension. The concatenated tensor become of size $[b \times 960 \times 60 \times 80]$. This concatenated tensor is similarly passed through *Convolution-A*, *Leaky ReLU-A*, *Convolution-B* and *Leaky ReLU-B* layers (see items 1, 2, 3 and 4 in Table. 3), where $\beta = \frac{F}{2} + 128$ and $\zeta = \frac{F}{4}$ for *Convolution-A* layer and $\beta = \frac{F}{4}$ and $\zeta = \frac{F}{4}$ for *Convolution-B* layer. From these operations, we will generate a tensor of $[t3 : b \times 416 \times 60 \times 80]$ dimension.

Then, the tensor $t3$ is again interpolated and concatenated with the output tensor from *Max Pooling Layer* (see item 5 in Table.1) i.e. the tensor of $[b \times 64 \times 120 \times 160]$ dimension. The concatenated tensor become of size $[b \times 480 \times 120 \times 160]$. This concatenated tensor is similarly passed through *Convolution-A*, *Leaky ReLU-A*, *Convolution-B* and *Leaky ReLU-B* layers (see items 1, 2, 3 and 4 in Table. 3), where $\beta = \frac{F}{4} + 64$ and $\zeta = \frac{F}{8}$ for *Convolution-A* layer and $\beta = \frac{F}{8}$ and $\zeta = \frac{F}{8}$ for *Convolution-B* layer. From these operations, we will generate a tensor of $[t4 : b \times 208 \times 120 \times 160]$ dimension.

After that, the tensor $t4$ is also interpolated and concatenated with the output tensor from *ReLU activation Layer* (see item 4 in Table.1) i.e. the tensor of $[b \times 64 \times 240 \times 320]$ dimension. The concatenated tensor become of size $[b \times 272 \times 240 \times 320]$. This concatenated tensor is similarly pass through *Convolution-A*, *Leaky ReLU-A*, *Convolution-B* and *Leaky ReLU-B* layers (see items 1, 2, 3 and 4 in Table. 3), where $\beta = \frac{F}{8} + 64$ and $\zeta = \frac{F}{16}$ for *Convolution-A* layer and $\beta = \frac{F}{16}$ and $\zeta = \frac{F}{16}$ for *Convolution-B* layer. From these operations, we will generate a tensor of $[t5 : b \times 104 \times 240 \times 320]$ dimension. Finally, this $t5$ tensor is passed through the following convolution layer where $\beta = \frac{F}{16}$

and $\zeta = 3$ to generate a tensor of $[t6 : b \times 3 \times 240 \times 320]$ dimension. The input im-

Convolution	No. of input filters/channels : β; No. of output filters/channels : ζ; Conv kernel : 3×3; Stride: 1×1; Padding: 1

ages are represented by their original colors in the range $[0, 1]$ without any input data normalization. Target depth maps are clipped to the range $[0.4, 10]$ in meters.

3 Further Details : Learning Loss Function

Technique-1 is explained in details in the main paper. In this section of supplementary materials, we will further describe *Technique-2* and its variants i.e. *Variant-1* and *Variant-2* of *Technique-2* in details.

3.1 Technique 1

As the first approach, the total loss (L_{total}) is computed by adding the loss over *depth* image i.e. L_d and the loss over *estimated simulated image* i.e. L_p. Hence, $L_{total} = L_d + L_p$. Where, L_d is computed by combining the weights λ_1^y and λ_2^y (see Equation (7)) and L_p is computed by combining the weights λ_1^q and λ_2^q (see Equation 10). These weights are initially set as 0.1.

3.1.1 Variant 1 : Instead of fixing the values of λ_1^q and λ_2^q as 0.1, we compute them automatically by using the same network, shown in Fig. 2. The pre-trained features (of size $[b \times 1664 \times 15 \times 20]$ from the last layer of the *DenseNet* based encoder i.e. from *Batch Norm - 5* layer (see item 13 in Table. 5) is branched out and passed through two consecutive *ConvBNRelu* blocks, shown in following Table. 4; where $\beta = 1664$, $\zeta = 512$, $k = 11$, $s = 1$ and $p = 1$ is taken for first *ConvBNRelu* block and $\beta = 512$, $\zeta = 256$, $k = 9$, $s = 1$ and $p = 1$ for the second *ConvBNRelu* block. Then the output feature is passed through the "Average Pooling" layer[1], having a kernel of size 1. This makes the feature to get transformed into $2D$ features which are then flattened and reduced into the $1D$ feature. Then these $1D$ features are passed through following layers:

DropOut-FC$(256 \rightarrow 128)$-**RELU**
DropOut-FC$(128 \rightarrow 64)$-**RELU**
DropOut-FC$(64 \rightarrow 32)$-**RELU**
DropOut-FC$(32 \rightarrow 16)$-**RELU**
FC$(16 \rightarrow 2)$

[1] here we have used "Adaptive Average Pooling" algorithm from PyTorch library. For more details, see : `https://pytorch.org/cppdocs/api/classtorch_1_1nn_1_1_adaptive_avg_pool1d.html`

After passing through the above layers, we will obtain two output values which are then passed through *Sigmoid* activation function to finally get two weight values. We apply this above mentioned technique to obtain two weight values i.e. w^1_{Depth} and w^2_{Depth} from the *encoder* of *Block-1* and another two weight values i.e. $w^1_{Residue}$ and $w^2_{Residue}$ from the encoder of *Block-2* (see Fig. 2 for reference).

Table 4: The *ConvBNRelu* block

Sl. No.	Layers	Specifications
1	Convolution	No. of input filters/channels : β; No. of output filters/channels : ζ; Conv kernel : $k \times k$; Stride: s; Padding: p
2	Batch Norm	No. of output filters/channels : ζ
2	ReLU	

Please note that here we have applied the *Sigmoid* activation function because the posterior probability values are between $0-1$ but the sum of these values can be greater than 1 (whereas, in the case of *SoftMax* activation function, the posterior probability values are between $0-1$ and the sum of all these values are 1)[2]. Hence, by applying *Sigmoid* activation function, we confirm that the individual weights i.e. w^1_{Depth}, w^2_{Depth}, $w^1_{Residue}$ and $w^2_{Residue}$ are within $0-1$. Moreover, by applying $(1-w^1_{Depth})$ or $(1-w^1_{Residue})$ on the second term for loss computation (see Equation 3), we confirm that only the remaining weight is applied on the second term. By using these automatic weight values, the L_d and L_p is calculated as:

$$\begin{aligned} L_d(y_p, \hat{y}_p) &= w^1_{Depth} L_{depth}(y_p, \hat{y}_p) + (1 - w^1_{Depth}) L_{SSIM}(y_p, \hat{y}_p) \\ L_p(q_p, \hat{q}_p) &= w^1_{Residue} \left[\frac{1}{n}\sum_p^n |q_p - \hat{q}_p|\right] + (1 - w^1_{Residue}) \left[\frac{1-SSIM(q_p,\hat{q}_p)}{2}\right] \\ L_{total} &= L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) \end{aligned} \tag{3}$$

3.1.2 Variant 2 : In addition with the weighted computation of $L_d(y_p, \hat{y}_p)$ and $L_p(q_p, \hat{q}_p)$ (according to Equation (12)), we also compute the total loss in the following manner, where the weight values w^2_{Depth} and $(1 - w^2_{Depth})$ are applied to perform weighted combination to the computation of total loss (i.e. L_{total}).

$$L_{total} = w^2_{Depth} L_d(y_p, \hat{y}_p) + (1 - w^2_{Depth}) L_p(q_p, \hat{q}_p) \tag{4}$$

3.2 Technique 2

As a second approach, a loss term corresponding to the "initial degraded" image (computed by using Equation (1) in the main paper) is added, compared to the total loss of Equation (12).

[2] For details, see : https://medium.com/arteos-ai/the-differences-between-sigmoid-and-softmax-activation-function-12adee8cf322

3.2.1 Variant 1 : In the same way as it is mentioned in Equation (12), here also we compute the weighted (automatic) version of $L_d(y, \hat{y})$ and $L_p(q, \hat{q})$. Whereas the weighted (automatic) version of $L_t(h, \hat{h})$ and the total loss is computed in the following manner. The needed weight w^2_{depth} is computed in the above defined manner.

$$\begin{aligned} L_t(h_p, \hat{h}_p) &= w^2_{Depth} \left[\frac{1}{n}\sum_p^n |h_p - \hat{h}_p|\right] + (1 - w^2_{Depth}) \left[\frac{1-SSIM(h_p, \hat{h}_p)}{2}\right] \\ L_{total} &= L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) + L_t(h_p, \hat{h}_p) \end{aligned} \tag{5}$$

3.2.2 Variant 2 : In the same manner, to compute the total loss, we need at-least 2 weight values. To obtain these weights, we use the same strategy as the one mentioned in Section 3.1.1. We take out another branch from the last layer of the *DenseNet* based encoder i.e. from *Batch Norm - 5* layer (see item 13 in Table. 5) and apply exactly same operation as before like passing through two consecutive *ConvBNRelu* blocks, shown in following Table. 4, applying "Adaptive Average Pooling" layer, followed by flattening operation which is followed by several blocks of **DropOut-FC-RELU**. But here, from the very last linear layer (i.e. **FC**$(16 \rightarrow 3)$), we obtain 3 output values which are then passed through *Soft-Max* activation function to finally get 3 weight values. Please note that here we have applied the *Soft-Max* activation function because the posterior probability values are between $0 - 1$ and the sum of all these values are 1. Hence, by applying *Sigmoid* activation function, we confirm that the individual weights i.e. w^1_{Extra}, w^2_{Extra} are within $0-1$. Moreover, by applying $[1-(w^1_{Extra}+w^2_{Extra})]$ on the third term for loss computation (see Equation 6), we confirm that only the remaining weight is applied on the third term.

$$\begin{aligned} L_{total} = w^1_{Extra} \times L_d(y_p, \hat{y}_p) + w^2_{Extra} \times L_p(q_p, \hat{q}_p) \\ + [1 - (w^1_{Extra} + w^2_{Extra})] \times L_t(h_p, \hat{h}_p) \end{aligned} \tag{6}$$

3.3 Technique 3

As the third modification, here we introduce an additional *encoder-decoder* block to directly estimate the simulated image ($I^{Direct}_{Predicted}$) from the clean *RGB* image. Hence, we compute a dedicated loss (L_g) for ($I^{Direct}_{Predicted}$) image only:

$$L_g(s_p, \hat{s}_p) = \lambda^s_1 \left[\frac{1}{n}\sum_p^n |s_p - \hat{s}_p|\right] + \lambda^s_2 \left[\frac{1 - SSIM(s_p, \hat{s}_p)}{2}\right] \tag{7}$$

where s_p and $\hat{s}_p$ represents the true and predicted "directly simulated" image ($I^{Degraded}_{Initial}$) by the network and the value of λ^s_1 and λ^s_2 are set to 0.1. Finally, the total loss is:

$$L_{total} = L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) + L_t(h_p, \hat{h}_p) + L_g(s_p, \hat{s}_p) \tag{8}$$

3.3.1 Variant 1 : As the *variant 1* under this category, in the same way as it is mentioned in Equation 16, here also we compute the weighted (automatic) version of

$L_g(s_p, \hat{s}_p)$ in addition to the weighted version of $L_d(y_p, \hat{y}_p)$, $L_p(q_p, \hat{q}_p)$ and $L_t(h_p, \hat{h}_p)$ (see Equation (9)) in the following manner. Here also the weight value of w^1_{Direct} is applied in the same manner to compute the weighted combination of total loss (L_{total}).

$$\begin{aligned} L_g(s_p, \hat{s}_p) &= w^1_{Direct} \left[\frac{1}{n}\sum_p^n |s_p - \hat{s}_p|\right] + (1 - w^1_{Direct}) \left[\frac{1-SSIM(s_p,\hat{s}_p)}{2}\right] \\ L_{total} &= L_d(y_p, \hat{y}_p) + L_p(q_p, \hat{q}_p) + L_t(h_p, \hat{h}_p) + L_g(s_p, \hat{s}_p) \end{aligned} \tag{9}$$

3.3.1.1 Variant 2 : As the *variant 2* under this category, in the same manner, here also we can compute the total loss in the following way. In this case, we need at-least 3 weight values which are obtained in the same manner as it is mentioned in Section 3.2.2.

$$\begin{aligned} L_{total} = w^1_{Extra} \times L_d(y_p, \hat{y}_p) + w^2_{Extra} \times L_p(q_p, \hat{q}_p) + w^3_{Extra} \\ \times L_t(h_p, \hat{h}_p) + [1 - (w^1_{Extra} + w^2_{Extra} + w^3_{Extra})] \times L_t(h_p, \hat{h}_p) \end{aligned} \tag{10}$$

One important thing to note here is that all the computed weight values ($\mathcal{W}$) e.g. $\mathcal{W} : w^1_{depth}, w^2_{depth}, w^1_{Residue}, w^2_{Residue}$ etc. computes weights based on a given RGB image. Hence, if there are b number of images in a batch then we will generate b number of such weights. But all these Equations for computing loss e.g. Equation (3), (4), (5), (6) etc. are directly computed on batches. Hence, to apply the weight values, we take it's mean over a batch i.e. $w = \frac{\sum_{n=1}^{b} \mathcal{W}_n}{b}$. This is also a reason to apply $1 - w$ amount of weights in the second term of the loss calculation equations e.g. Equation (3), (4) etc.

Table 5: The number of trainable parameters of each block for all the proposed techniques. The time needed to execute each epoch for each of the techniques. The extra parameters in *Variant-1* and *Variant-2* of each block appears due to supplementary branch, needed for the automatic weight computation.

Proposed Method	Block-1	Block-2	Block-3	Training Time
Technique-1	443, 22, 689	443, 24, 563	×	2 hr 39 min 25 sec
Technique-1 Variant-1	1580, 73, 747	1580, 75, 621	×	3 hr 21 min 35 sec
Technique-1 Variant-2	1580, 73, 747	1580, 75, 621	×	3 hr 28 min 19 sec
Technique-2	443, 22, 689	443, 22, 689	×	2 hr 26 min 56 sec
Technique-2 Variant-1	1580, 73, 747	1580, 75, 621	×	2 hr 49 min 17 sec
Technique-2 Variant-2	1580, 73, 747	1580, 75, 621	×	2 hr 49 min 18 sec
Technique-3	443, 22, 689	443, 24, 563	443, 24, 563	5 hr 41 min 59 sec
Technique-3 Variant-1	1580, 73, 747	1580, 75, 621	1580, 75, 621	5 hr 59 min 24 sec
Technique-3 Variant-2	1581, 17, 558	1580, 75, 621	1580, 75, 621	5 hr 59 min 27 sec

4 Training Details

For each of the different proposed network configurations, the total number of trainable parameters are mentioned in Table. 5. We have also mentioned the time needed to execute one epoch for each of the proposed techniques. It can be seen that *Technique-3* and

two of its variants have a lot more training parameters and it also takes more time to train. The training parameters are mentioned in Table. 6. We have used *Adam Optimizer* from PyTorch[3] library for the training. Except learning rate, we have taken the default values of other parameters.

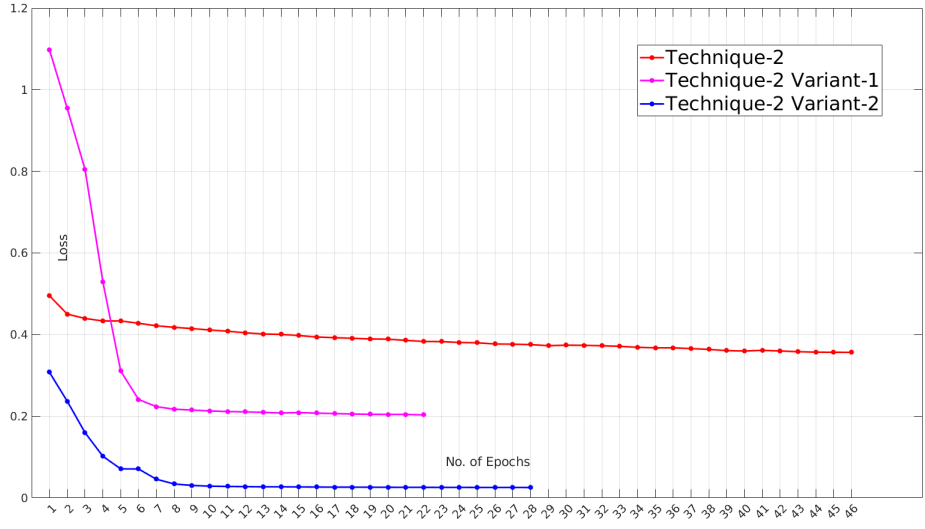
Table 6: The parameters for training

Proposed Method	Optimizer	Learning Rate
Technique-1	Adam	$1e^{-6}$
Technique-1 Variant-1	Adam	$1e^{-6}$
Technique-1 Variant-2	Adam	$1e^{-6}$
Technique-2	Adam	$1e^{-6}$
Technique-2 Variant-1	Adam	$1e^{-6}$
Technique-2 Variant-2	Adam	$1e^{-6}$
Technique-3	Adam	$1e^{-5}$
Technique-3 Variant-1	Adam	$1e^{-5}$
Technique-3 Variant-2	Adam	$1e^{-6}$

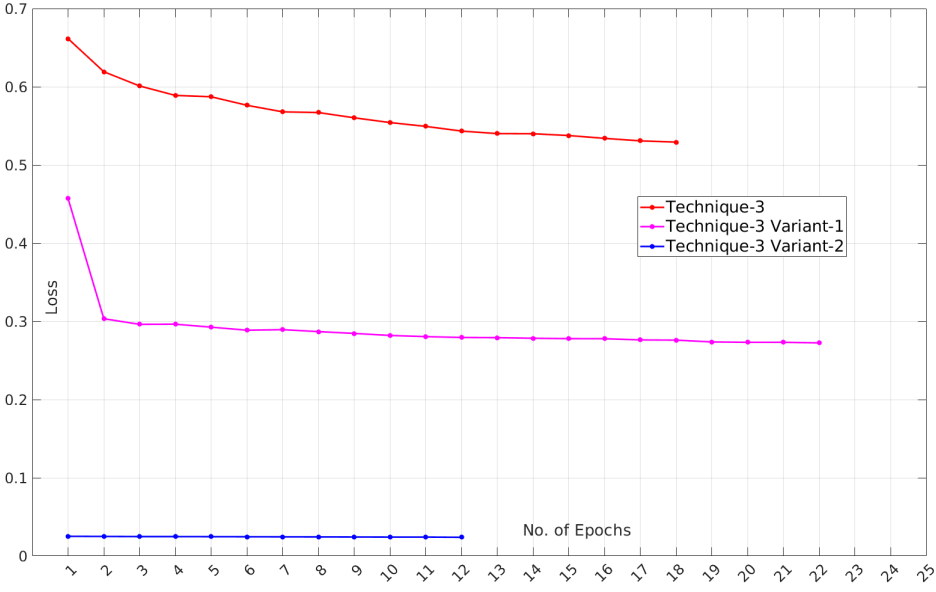
4.1 Training Loss

The training loss plots of the *Technique-2*, *Technique-2 Variant-1*, *Technique-2 Variant-2* and consecutively the training loss of *Technique-3*, *Technique-3 Variant-1*, *Technique-3 Variant-2* are shown in Fig. 2a and Fig. 2b respectively. It can be visible from these plots that the training losses of all the techniques decreases gradually whereas the *Variant-1* and *Variant-2* of both the *Technique-2* and *Technique-3* gets stabilized quickly (i.e. less number of epochs are needed). Moreover, we can also see that either training loss decreases rapidly (see the plots of *Technique-2 Variant-1*, *Technique-2 Variant-2* and *Technique-3 Variant-1*) or it starts with low value and stabilizes quickly (see the plot of *Technique-3 Variant-1*). Please note that *Technique-1* along with it's variants has similar characteristics and performance as *Technique-2* and it's variants, here we have shown the loss curve on *Technique-2* and it's variants only.

[3] `https://pytorch.org/docs/stable/generated/torch.optim.Adam.html`



(a)



(b)

Fig. 2: (a) The training loss curve of *Technique-2* and it's variants (b) The training loss curve of *Technique-3* and it's variants

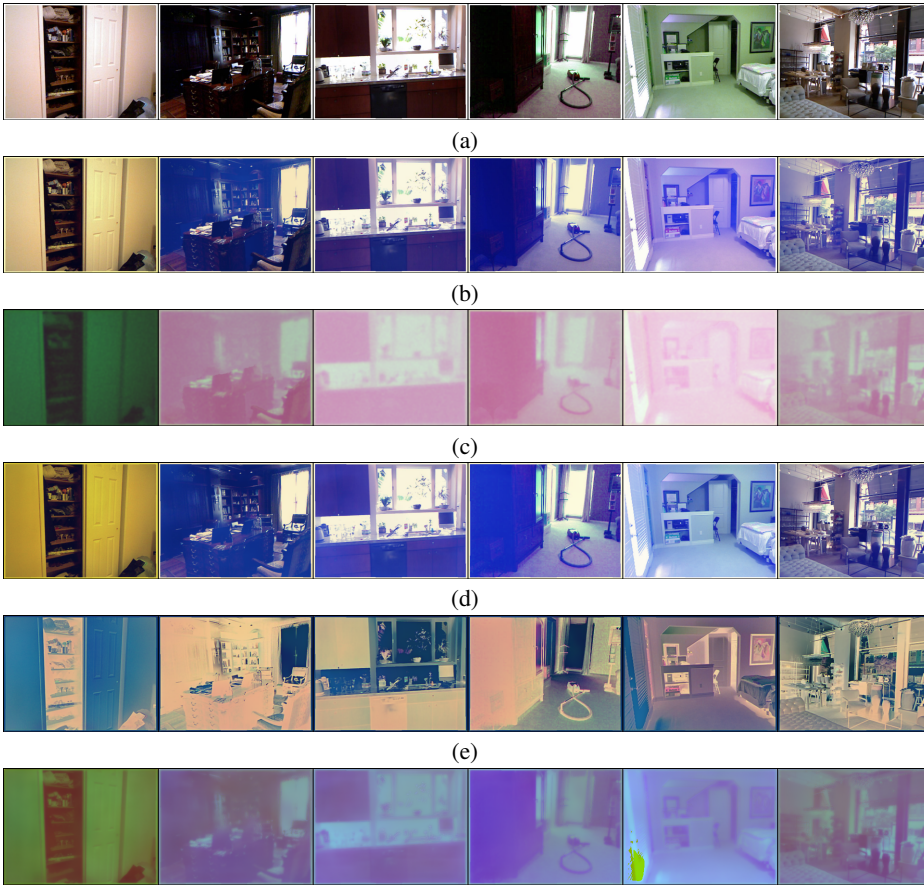


Fig. 3: **Qualitative Measures obtained by *Proposed* Technique 1 :** (a) Original *RGB* images (i.e. $J_c(\mathbf{x})$) (b) Ground Truth of "Initial Degraded Image" (i.e. $I_c(\mathbf{x})$) (c) Ground Truth of "Simulated Underwater Image" (d) Predicted "Initial Degraded Image" (i.e. $I_{Initial}^{Degraded}$) (e) Predicted "Residual Image" (i.e. $I^{Residue}$) (f) Predicted "Simulated Underwater Image" (i.e. $\hat{I}_{Predicted}^{Simulated}$).

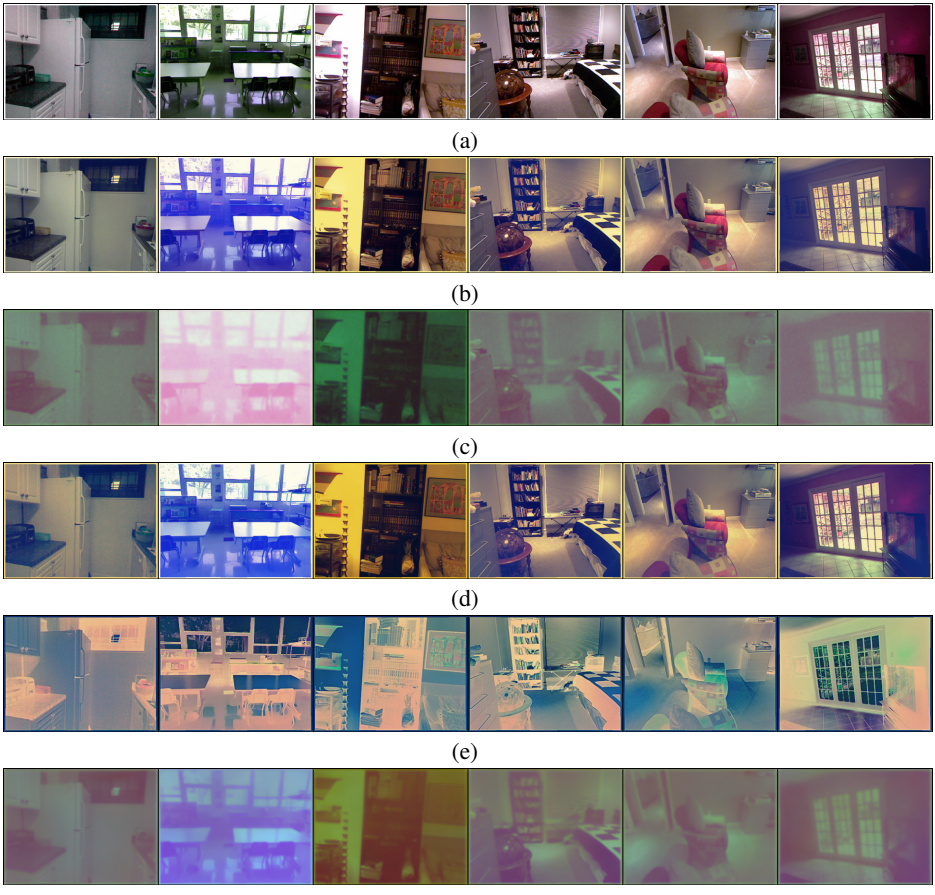


Fig. 4: **Qualitative Measures obtained by *Proposed* Technique 1 Variant 1 :** (a) Original *RGB* images (i.e. $J_c(\mathbf{x})$) (b) Ground Truth of "Initial Degraded Image" (i.e. $I_c(\mathbf{x})$) (c) Ground Truth of "Simulated Underwater Image" (d) Predicted "Initial Degraded Image" (i.e. $I_{Initial}^{Degraded}$) (e) Predicted "Residual Image" (i.e. $I^{Residue}$) (f) Predicted "Simulated Underwater Image" (i.e. $\hat{I}_{Predicted}^{Simulated}$).