UNDERWATER IMAGE RESTORATION

Sandeep Thalapanane UID – 119402535
Sourang SriHari UID – 119074947
Aaqib Barodawala UID – 119348710
Sandip Sharan Senthil Kumar UID – 119340196
Shreejay Badshah UID – 119224564

Abstract

The human era is witnessing the most sophisticated period of computer vision and with the help of artificial intelligence and machine learning, functionalities like image recovery, object detection, defect analysis and much more can be accomplished in no time. Despite the availability of the plethora of resources and new findings, one area of computer vision remains untapped: Underwater Image Restoration.

Restoring underwater images is useful in many practical applications such as analyzing marine ecosystems, identifying plant and animal species, underwater inspection and maintenance, photography, and filmmaking. The research papers for the past 5 decades have been following the conventional protocol or methodology for restoring images of underwater bodies.

Introduction

Image restoration within the underwater domain is governed by the equation:

$$I_C = D_C + B_C \qquad (1)$$

The above equation is the formula that oversees the image processing for underwater objects. Here I_C is the image captured by the camera, D_C is the direct signal that gives information about the attenuated image, and B_C is the term that governs backscatter of the respective images, also known as the veiling light. The subscript "c" represents the color channel in RGB space.

Expanding equation (1), we obtain:

$$I_C = J_C e^{-\beta_C \cdot z} + B_C^{\infty}(1 - e^{-\beta_C \cdot z}) \qquad (2)$$

where z is the distance of the object from the camera,
$e^{-\beta_C \cdot z}$ is the attenuation coefficient,
B_C^{∞} is the veiling light,
J_C is the unattenuated scene captured by the camera if there was no attenuation along z.

The problem arises with utilizing the attenuation coefficient for both the direct signal and the backscatter component. The backscatter component or the veiling light, which arises due to the scattering of light due to tiny particles in water, comprises of the coefficient that oversees the augmentation of the backscatter which is completely distinct from the attenuation coefficient. Numerous errors occurred due to considering the same attenuation coefficient for the direct signal and the veiling light/backscattered signal. Another problem is that the attenuation coefficient is considered constant for all depths and reflectance which is not the actual case for practical imaging (as it drastically changes for different depths and for different types of water e.g., Mediterranean Sea, Red Sea, etc.).

The updated equation from (2) is given by:

$$I_C = J_C e^{-\beta_C^D(v_D).z} + B_C^{\infty}(1 - e^{-\beta_C^B(v_B).z}) \qquad (3)$$

where β_C^D is the attenuation coefficient,
β_C^B is the backscatter coefficient,
$v_D = \{z, \rho, E, S_c, \beta\}$ and $v_B = \{E, S_c, b, \beta\}$

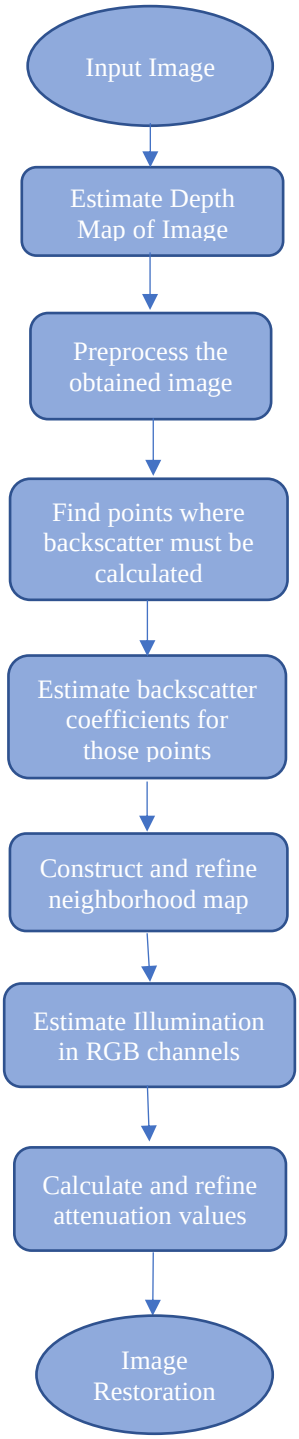
The v_D and v_B vectors represent the dependencies on coefficients β_C^D and β_C^B on range z, reflectance ρ, spectrum of ambient light E, spectral response of the camera S_c, beam attenuation coefficients of the water body b and β. All of the above dependencies are functions of wavelength.

Previously, it was assumed that $\beta_C^D = \beta_C^B$ and they had the same single value for a given scene. However, this was proven to be distinct, and they have various dependencies based on different factors.

It is shown that β_C^D heavily depends on z whereas β_C^B is most affected by the optical water type and illumination E. As a result, the method mentioned in this paper utilizes a depth estimation model using a deep neural network architecture to predict depth of the scene/object.

Methodology

1. Pipeline:



The algorithm takes input from the user and the image can be in any format. Then the obtained image is resized as per the size requirements of the DNN model to predict the depth map for the image. Then the user input image is downscaled. Then the obtained depth image from the model is pre-processed by taking its upper and lower limits by assigning the values in depth image lesser than the lower bounds to 0 and the values which are zero to the upper bound. Then, points where the backscatter values are to be calculated by partitioning the image into different depth ranges and taking the darkest RGB triplets from the set as estimation points of the backscatter. Then, calculate the backscatter coefficients based on the backscatter point values and their respective depth values using the equation given below.

$$\hat{B}_c = B_c^{\infty}(1 - e^{-\beta_c^B z}) + J_c^{'} e^{-\beta_c^{D'} z} \quad (4)$$

Then, find the best coefficient values by optimizing the coefficients through curve fitting and calculating the loss values for each set of coefficients. Then, using the depth image and epsilon value construct a neighborhood map by grouping the similar points of the input image, specifically based on the depth value. After constructing the neighbor map, refine it to remove unnecessary artifacts. Then, estimate illumination map from range map using local space average color (LSAC). Briefly, illuminant map is used to calculate the average color value for a small area which is given by the neighborhood map computed previously. To do this, just calculate the local space average color for a certain pixel (x, y) in color channel c, $a_c(x, y)$ is estimated iteratively through updating the equations:

$$a_c'(x,y) = \frac{1}{N_e} \sum_{N_e} a_c(x', y') \quad (5)$$

$$a_c(x,y) = D_c(x,y)p + a_c'(x,y)(1-p)$$

where the neighborhood N_e is defined as the 4-connected pixels neighboring the pixel at (x, y) which are closer to it than a range threshold $\in$:

$$N_e(x', y') = (x', y') \quad \text{with } \| z(x, y) - z(x', y') \| \leq \in \quad (6)$$

Next, using an optimization framework, estimate an approximate value the range-dependent attenuation

coefficient (β_d) using an illumination map as input. Next, refine the estimated value of (β_d by calculating reconstructed depth values and find the best β_d value by comparing it with the loss value given by taking the mean of original depth value for that pixel and subtracting it from the calculated reconstructed depth value. Then using the calculated value of β_d and β_c we can restore the underwater image which is explained in Section **(4)**.

2. Depth Estimation using Deep Learning:

The main motivation to create a depth estimation model is to eliminate the help of stereo cameras and the need for depth maps to restore the images. To estimate the depth of an underwater image a machine learning model is used to predict the depth using a monocular image. The model is trained with images and its corresponding depth maps using a neural network and after the training process, the model is used to create depth maps from monocular images.

The depth estimation of an underwater image is a tricky process, and the model needs to be trained on an underwater dataset that contains underwater images and their corresponding depth maps. There are only two underwater datasets available on the internet with images and depth maps. Unfortunately, both datasets could not be downloaded due to some link expiry issues. So, the model was trained on a dataset with images which are taken on the land.

The libraries used in the model and for the training are TensorFlow, Pandas, Keras, and OpenCV. The following process is preparing the data frame of the model which primarily segregates the image, depth and mask elements using its respective formats (.png and. npy formats and labels). The size of each image is 256 x 256. The next and very important step involves describing the hyperparameters of the model. The metrics that govern the hyperparameters of the model are learning rate (0.0002), Epochs (30) and batch size (16).

The height and width of the hyperparameters limit the size of all the images used in the dataset, the batch size giving information about how many images will be embedded per batch, epochs being the number of times or cycles the model has been trained using the training data that we retrieved from above, and finally the learning rate which is basically the step size at every iteration that converges towards the cost of the model.

The data frame which comprises all three formats of files including the images, depth mask and depth files is embedded into this pipeline and the following functions are performed on it: Initially, image processing is done on the images and are uniformly resized in terms of shape and format throughout the dataset. Then the depth and depth mask files are processed so that we obtain the image format of the depth map, which is again resized.

The class Generator data in the data pipeline prepares the batch size, number of epochs, and shuffling of the training data set so that the model learns well from the same dataset through various distortions and changes. The remaining code focuses on resizing and formatting the images uniformly. The next step is data visualization of a few random samples where the original images which have been formatted are displayed on the left-hand side and the depth map is used on the right-hand side of the screen. The depth map is an image with a heat map, indicating different depths of objects within the images.

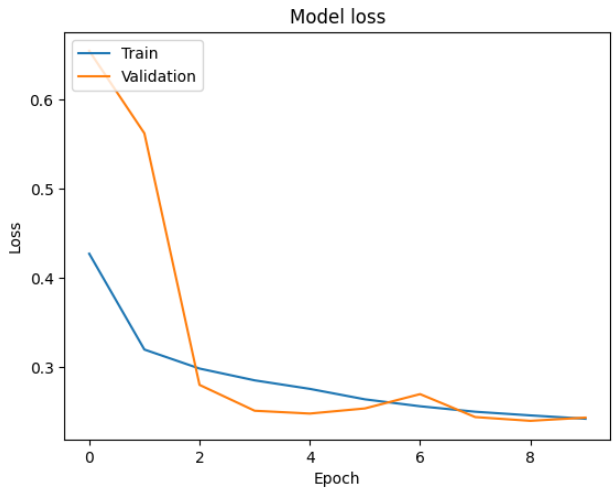
After that, the focus is on building the network model for the data that has been preprocessed in the earlier step. The model construction is done in the following way: Initially, the data is downscaled, reaches a bottleneck, and then increased, augmented or upscaled to project the data in higher dimensions. Initially, a simple convolutional layer with an activation function ReLU and batch normalization layer is built for each upscaling and downscaling network which is inspired by the U-Net.

The downscaling (encoder) part comprises of 3 convolutional layers, followed by 3 batch normalization layers. It is then followed by a batch of 3 ReLU activation layers and finally finishing the downscaling architecture with a Maxpooling layer to completely downscale the input features. Following the downscaling phase is the bottleneck layer, which is the bridge between downscaling and upscaling. This layer comprises 2 sets of convolutional layers and ReLU activation layers. The upscaling (decoder)

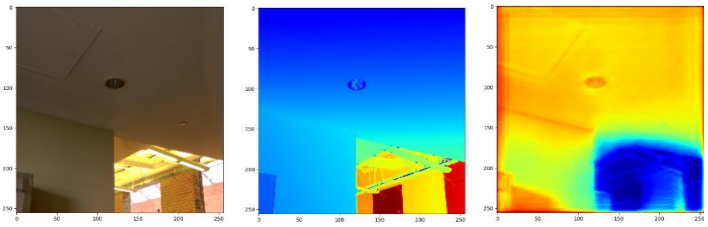
architecture contains a similar one to that downscaling, with an extra upscaling layer and a concatenation layer precluding other layers.

The loss functions used in the model are the Structural similarity index (SSIM), L1-loss, or Pointwise depth and Depth smoothness loss. Out of the three loss functions, SSIM contributes the most to improving model performance. These loss functions are embedded into all three blocks of the model construction including downscaled, upscaled and the bottleneck respectively. The gradient descent is applied to the SSIM loss function, since the gradient descent is used to this loss function it has a high priority among the three. The final step involves training and testing the model and the results are obtained accordingly.

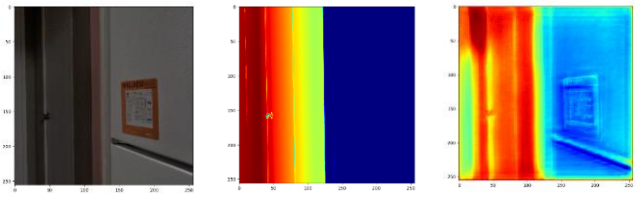
The image below shows the training and validation loss for 9 epochs:



The depth map estimate results for the DIODE dataset are given below:



Actual Image Predicted Depth map Actual Depth map



Actual Image Predicted Depth map Actual Depth map

3. Incorporating Deep Learning Model:

Integrating custom Deep Neural Network (DNN) model into an image color correction algorithm presents a promising opportunity to enhance the acquisition of depth images. Using the saved model, incorporated that model with the image correction algorithm to get a depth image. For this, first load the model using TensorFlow by giving the path of the saved model. Then, by harnessing the capabilities of deep learning, our DNN model can learn intricate color patterns and intricate relationships, empowering accurate and reliable depth estimation.

The algorithm commences by feeding an RGB image into a data generation pipeline where the image dimension gets altered according to the batch size set by the user. Then, that updated image data is sent to the DNN model, which subsequently undergoes a series of convolutional and fully connected layers. These layers extract essential features and acquire complex mappings between color information and corresponding depth values. Through extensive training, our DNN model effectively captures the inherent connections between color and depth, enabling precise depth predictions for each pixel in the image. This amalgamation of our custom DNN model and the color correction algorithm unlocks the potential for generating superior quality depth images, applicable across diverse domains such as 3D reconstruction, augmented reality, and comprehensive scene comprehension.

4. Image Restoration:

Using the revised image formation model, after getting the values of β_d *and* β_c the image can be restored and even enhanced by a method based on Gray World Hypothesis where we perform scene reconstruction and global white balance for the input image.

4.1 Scene Reconstruction:

First, we calculate direct signal value by subtracting the updated illumination values (B_c) from the original input image. It is then multiplied with the exponential of updated depth coefficient values (β_d) and expanded to match the dimensions of the image. The final restoration is obtained by element-wise multiplication of the depth-adjusted image and the illumination-corrected image. Finally, the restoration values are clipped between 0.0 and 1.0. The equation below, sums up the process of scene reconstruction:

$$D_c = I_c - B_c \quad (7)$$

$$J_c = D_c e^{\beta_c^D(z)z} \quad (8)$$

In the above equation J_c is the unattenuated scene of the image given by the user which will the perfect reconstructed image after removal of the water effects from the image. Then the reconstructed image is sent for color balancing.

4.2 Image scaling and white balancing:

The restoration values corresponding to background regions are set to 0, effectively removing any influence of the background on the restored image. Then, the image balancing is performed on the image by selecting top 10% of those values and then calculate the reciprocal of its average value which the scaling factor required for adjusting the channel values. The scaling values are doubled to increase their effectiveness on the image restoration. Overall, this process aims to achieve white balance by amplifying or attenuating specific color channels to bring the overall color cast of the image closer to neutral.

Then the resultant image is sent to another function where we apply histogram equalization to enhance the contrast of the image by assigning clip limit. Then, the noise in that image is reduced using a technique based on minimizing the total variation of that image. These steps can be beneficial in various image processing applications where contrast enhancement and noise reduction are desired.

Dataset

The dataset which we used to train the model was DIODE: A Dense Indoor and Outdoor Depth Dataset. DIODE contains RGBD images of outdoor and indoor scenes. There are two partitions in the dataset which are Train and validation. The size of the Train DIODE dataset was 81GB which makes it too difficult and time taking process to train the model. The other partition validation comprises only 2GB, and it was used to train the model. The validation was further split into indoor and outdoor scenarios. The outdoor image scenes are selected to train the model so that it performs better than the indoor scenarios for the underwater image depth estimation.

Results

Fig. 1: Original Image

Fig. 2: Rescaled Image

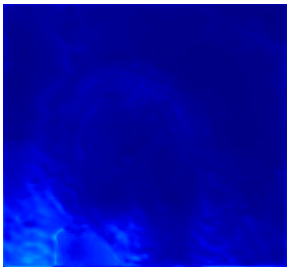


Fig. 3: Depth Map

Fig. 4: Restored Image

Figure 1 is the actual representation of the image underground. Note that the picture experiences extreme backscatter and is not the actual color representation of the objects we have seen in the image.

Figure 2 is the depiction of the image after downscaling is done. The function of downscaling is to reduce the dimensionality of the image and scale it down to quantifiable entities for the purpose of process. The images are initially in the form of 250 pixels lengthwise and breadthwise which are broken

down into feature maps at the end or output of downscaling. Then these feature maps are utilized to perform depth estimation using which the depth map for these images are generated.

Figure 3 shows the depth of map generated from the actual set of images. Finally, the image is restored with actual colors from the output of this depth map generated.

Figure 4 shows how these underwater objects look in the outside world and how much the backscatter has diminished the quality of these images, as seen in Figure 1.

Another result for a different scene is given below:

Fig. 1: Original Image

Fig. 2: Rescaled Image

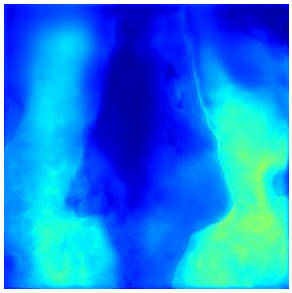


Fig. 3: Depth Map

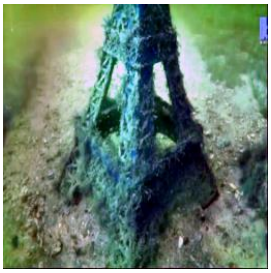


Fig. 4: Restored Image

Github Link

https://github.com/sandipsharan/Underwater-Image-Restoration.git

Conclusion and Future Work

From the results, regardless of using an above ground dataset for training our model, the algorithm removes water from the images and restores colors of objects depending on the estimated depth. This restoration can be further improved by training the model on underwater datasets.

Future work would involve training the model on underwater datasets which have raw images and the corresponding depth maps available. This would train the model to be more robust and precise. Moreover, the pipeline used downscales the image to 256×256 pixels which is not practically applicable for correctness. This will be rectified for future work so that algorithm works without rescaling. Lastly, real-time implementation would be another aspect to improve on for this project.

With these improvements, the practical applications would be limitless. Some of the applications will be to boost underwater research to study marine biology, structural analysis and maintenance, deep sea exploration, etc.

Challenges Faced

There are several challenges that were faced while training the model, incorporating the model into the algorithm and the algorithm. The challenges faced were listed below:

1) Gradient explosion – Since the loss was converging to infinity, the model was not training properly. To fix this issue L2 regularizes were added, a proper weight initializer ‘He normal’ initializer was used in the model, and Adam optimizer was added along with clip norm value 1 to avoid gradient explosion. The activation function was changed from ReLU to Leaky ReLU to increase the model performance.

2) Decreasing Loss – The loss was in the range of 0.3 when the model was first trained and to decrease the loss furthermore convolution layers were added in the network.

3) Overfitting – The model was overfitting after running for 20 epochs and to overcome the problem an early stopping call back has been added to the model to stop the training when the train loss and validation loss are the same.

4) Incorporating DNN - While incorporating the DNN model, the model required input data in a specific dimension, (i.e.,) (number of batches, image size with number of channels). In this case, the model requires just only one image input

therefore the data has been generated from that image by extending another dimension with empty values in it and then it is sent to the DNN model to generate the depth image. Another challenge was using a depth colormap which resulted in many problems as the number of channels increased to 4. So, the depth map given by the model was grayscale and was working fine. After running the comparison test between using colormap and grayscale map, both gave the same result so there was no drawback in switching back to grayscale.

5) Parameters tuning - Tuning various parameters such as epsilon, locality control parameter (p), balance control parameter(l) as the model wasn't trained on data from underwater images. Tuning the parameters given by the author to restore the image using the depth image given by the custom model was a difficult task.

Contributions

Team Members	Underwater Restoration Algorithm	DNN Model
Sandeep Thalapanane		X
Sandip Sharan Senthil Kumar	X	
Shreejay Badshah	X	
Sourang SriHari		X
Aaqib Barodawala	X	

References

[1] D. Akkaynak and T. Treibitz, "Sea-Thru: A Method for Removing Water From Underwater Images," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019.

[2] D. Akkaynak and T. Treibitz. A revised underwater image formation model. In *Proc. IEEE CVPR*,2018.

[3] Vasiljevic, Igor & Kolkin, Nick & Zhang, Shanyi & Luo, Ruotian & Wang, Haochen & Dai, Falcon & Daniele, Andrea & Mostajabi, Mohammadreza & Basart, Steven & Walter, Matthew & Shakhnarovich, Gregory. (2019). DIODE: A Dense Indoor and Outdoor DEpth Dataset.

[4] Ronneberger, Olaf, Philipp Fischer and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." ArXiv abs/1505.04597 (2015): n. pag.