CMSC828L PROJECT

UNDERWATER SLAM

BY :

ANIRUDH NAKRA

HENOK GEBEYEHU

WEI YU CHEN

CMSC 828L: Deep Learning
Project: Underwater SLAM

Anirudh Nakra, Henok Gebeyehu, Wei Yu Chen

University of Maryland,
College Park
Prof. David Jacobs

May 3, 2022

Abstract

Underwater robotics is a challenging area. Navigation and other applications have been concentrated to above ground environments. As a result, there has been very little work done on the behaviour of vision systems underwater. In this project, we extend and compare the state of underwater robot navigation with the focus being on having a thorough evaluation for hue correction and enhancement. Specifically, we choose to explore a self created Blender data set as well as a field test video acquired by us to study selective attenuation based physics models as well as interesting generative models based on rigorous mathematical techniques such as Wavelet decomposition. Later in the project, we perform different SLAM implementations on the hue corrected videos and analyse our findings through experiments as well as some mathematical lemmas. In this way we hope to encapsulate the world of Underwater navigation and robotics. [1]

1 Introduction

Computer vision underwater is a challenge. Due to the nature of water itself doing vision under controlled underwater environments have proven to be a monumental task, let alone natural water masses which can be categorized differently according to temperature, pressure, turbidity...etc. Each with vastly different characteristics which can have vastly different effects on light passing through. Therefore, to focus down on the plethora of different problems we will focus on a relatively more popular environment for this field of research, which is underwater image color correction for robot SLAM within a depth of 10 meters with the water being not turbid or mildly turbid.

This paper is about comparing methods for underwater image color restoration and the effects it has on SLAM systems. Namely ORBSLAM3[2] and DSO[4]. Therefore to narrow down the scope even further we took data at ideal condtions, one dataset being a simulation on Blender and the other being taken from the Neutral Bouyancy Research Facility at UMD. These datasets are mostly taken to minimize multiple effects that if taken consideration would take us out of scope. Therefore in the environments we took our data in we don't need to take into consideration the different temperatures, pressure, current...etc that would otherwise overcomplicate things and focus mainly on color correction and the little bit of turbidity and bubbles that would always persist underwater.

2 Past Works

2.1 The Jaffe-McGlamory model

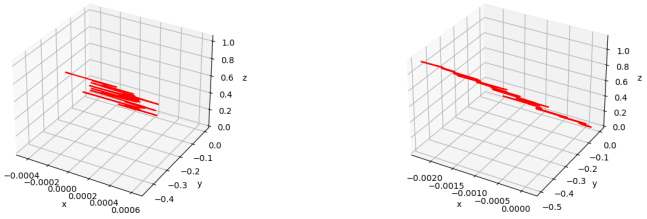
The Jaffe-McGlamory model is discussed in [1],[4] and [6]. Since we know that regular in-air vision is relatively well researched and most vision methods are designed based on it, naturally the quickest way to deal with underwater vision is to try to modify the image to the point where in-air methods would yield acceptable results. The Jaffe-McGlamory model is an underwater imaging model which modifies in-air images to look like underwater images by modeling the disturbances into forward scattering and back scattering. Where we get equation

$$E_T = E_f + E_b \tag{1}$$

which describes the fact that the light that goes into the camera E_T is a combination of E_f which is the light from forward scattering, and E_b which is the light from back scattering. Forward scattering is also known as attenuation, which is the effect when the saturation and brightness of the image is lowered with respect to the distance underwater, which is the main reason why you can't see the other side of a swimming pool if you dive underwater where it is easily visible when you surface, this is modeled by

$$E_f = I_{in} e^{-\eta(\lambda) r} \tag{2}$$

[1] Our experiments can be accessed at this Drive link



(a) unprocessed (b) Wavelet-GAN

Figure 1: Here we see that on the synthetic data the unprocessed images does decently regarding the y and z axis but is completely inaccurate on the x axis in a seemingly random way. Whereas images from Wavelet-GAN still does decently in the y and z axis and the randomness on the x axis is significantly reduced.

where I_{in} is the in-air input image, $\eta(\lambda)$ is the attenuation coefficient which consists of the sum of the absorption coefficient a_λ and scattering coefficient b_λ, all of which is dependent on the wavelength (In the case of digital images since we deal with RGB images the wavelength could also be represented as separate channels), r is the distance between the source and the camera. Back scattering is in general the effect when light reflects off unwanted particles such as sand particles, plankton, or bubbles. Which creates the hazy feel in underwater images and is modeled by

$$E_b = \beta(\lambda)(1 - e^{-\eta(\lambda)r}) \tag{3}$$

where $\beta(\lambda)$ is an added backscattering light with respect to the wavelength (which in most approaches are treating as a random noise). Therefore, we can derive the underwater image model as

$$I_{out} = I_{in}e^{-\eta(\lambda)r} + \beta(\lambda)(1 - e^{-\eta(\lambda)r}) \tag{4}$$

which would give us the restoration model we desire as

$$I_{in} = \frac{I_{out} - \beta(\lambda)}{e^{-\eta(\lambda)r}} + \beta(\lambda) \tag{5}$$

and allows us to either train models or utilize image processing methods to attempt to solve for $\beta(\lambda)$,$\eta(\lambda)$, and r.

2.2 WaterGAN[5]

One approach that is previously proposed and is widely cited in this field of research is WaterGAN. WaterGAN is a GAN network for generating realistic underwater images that have similar properties, it utilizes a generator that contains three main parts. The first two parts are variations of the Jaffe-Mcglamory model using a RGBD input along with a 100-length noise vector to generate and reshape to a 48x64x3 synthetic image, then it adopts a $V = 1 + ar^2 + br^4 + cr^6$ vignette to the generator as the third part before using a discriminator to classify synthetic images from a real image.

Since the main part of WaterGAN is to generate underwater images from in-air images, it only gives a brief statement on approach for image restoration. Involving 2 SegNets. The first is a depth estimation network which is trained to estimate the depth image of the image using the synthetic underwater image and the in-air depth map. The second is the color restoration network which is trained using the synthetic RGBD image to the original in-air image. After the model is sufficiently trained the real underwater data could then be tested on this model.

Therefore, we ponder the possibility of methods, which includes deep learning and image processing methods, that could do color correction with only real underwater RGB images as input and see how it affects the performance of underwater SLAM.

3 Datasets

3.1 Blender Data

This dataset is a sequence of images in time totaling up to 500 images that basically moves in a straight line forward. Since this dataset is taken in the most ideal environment possible and the trajectory being very simple, we doubt that this would be a good indication that an algorithm that performs well on this dataset would also perform well in the real world. However, what this dataset does is it allows us to get a general idea of the feasibility the algorithm provides, as demonstrated in Figure 1. Also the camera intrinsic parameters would be well available since this does not require calibration and the focal length would only affect the sense of distance of SLAM, which in monocular SLAM the distance given are all relative distance so the focal length here doesn't matter the slightest. Therefore, since the images are 640×480 pixels we defined the camera matrix and distortion coeffiecients(used in OpenCV) as

$$M = \begin{bmatrix} 540 & 0 & 320 \\ 0 & 540 & 240 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \tag{7}$$

3.2 NBRF Data

This dataset is a video shot at 30 frames per second from the front camera of a Bluerov2 underwater robot at about 15 meters depth(which is the max depth at the NBRF), totaling at 838 frames. Even though this dataset is still taken within a relatively ideal environment, there exists some turbidity in the water from particles at the bottom of the pool. Also, the trajectory is much trickier since the robot is driven by hand. Therefore this is a pretty challenging trajectory to map since it involves irregular yaw movement and the trajectory doesn't overlap. We would consider a method that works well on this dataset would perform decently on underwater robots that are tasked in similar environments such as a swimming pool service robot or a fish farm cleaning robot. The camera here is calibrated using OpenCV cv2.calibrateCamera() function. The camera used is shot at 1920×1080 pixels and the camera matrix and distortion coefficients to be

$$M = \begin{bmatrix} 2553.77467 & 0 & 875.160353 \\ 0 & 2531.47073 & 441.398360 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$D = \begin{bmatrix} -0.07648 \\ 12.029 \\ -0.01001 \\ -0.01924 \\ -81.616 \end{bmatrix}^T \tag{9}$$

4 Image Processing Methods

Image processing is an important area to consider for this problem. This is due to the interpretability of the techniques applied and the fact that these techniques have great mathematical rigor in them. This conceptual way of looking at things is not to be taken lightly and these methods must be studied alongside deep learning techniques which are performance focused to help develop theoretical foundations to deep learning based techniques themselves! A lot of vision involves the use of convolutional models but it is still a mystery in the scientific world as to what exactly the filters of the CNNs learn. The first few layers are simplistic edge detectors but it is harder to interpret what happens at deeper layers. Although this interpretability is not of extreme importance when developing ML algorithms for specific tasks and they can be considered black boxes, it is an important issue to consider when performing comparative analysis or really writing any real research since it might provide insights to future researchers.

4.1 Fundamental methods

The first technique I implemented is perhaps the most fundamental technique in image processing. I implement Histogram Equalization. Consider an image whose pixel values are confined to a specific dynamic range/ range of intensity values. It has been empirically found that brighter images have histograms with a higher dynamic range as well as the pixels being more uniformly spread out when compared with dimmer, low contrast images that have their pixels clumped up in a small region in the histogram. Therefore, there is a need to perform an operation to stretch this concentrated histogram to a more uniform one and this is the intuition behind histogram equalization. This therefore helps to increase the contrast of the image.

This method is particularly useful in cases where both the background and foreground are dim or bright. Some of the most common applications include X-ray photography in which usually the bine structure is either over or under exposed and Histogram equalization can help lead to a better view in this use case. Histogram equalization is also advantageous due to the fact that it is not very computationally expensive. In fact, the method is real time and can even be designed to be invertible if the mapping function is known. Furthermore, this technique is adaptive to the histogram of the original image which is an important desirable property. Since the process is not adaptive to the scene object, it might end up increasing the noise contrast along with the contrast of the object in focus resulting in even more problems and a worse, more corrupted image and therefore there needs to be care on how and where this technique is applied.

A similar technique as Histogram Equalization is called Contrast stretching. This technique is less harsh on the image than Histogram equalization. This is due to the fact that contrast stretching only applies linear scaling function to the image pixel values and therefore is not as sophisticated as histogram equalization. This can be thought of as a linear normalization and is perhaps one of the more simpler methods that can be used. Although this technique uses a grayscale image by default, multiple different implementations working with RGB and other color spaces have also been worked upon. Given is the widely used histogram equalization transformation function.

$$T(x) = (L-1) \cdot \int_0^x p_1(w)\ dw,$$

Changing color spaces is also one of the more fundamental ideas in image processing. Several color spaces exist, RGB, HSI, HSL and YCbCr are some of them. Out of these numerous color space based techniques some stand out for their ability to capture and disentangle some sorts of information. For example, HSV successfully disentangles the chromaticity of the image (represented by the Hue channel) and the Intensity pat-

terns. YCbCr also takes a similar approach but represents the chromaticity using multiple channels rather than HSV which uses one channel for the chromaticity and two for lighting information. There are a lot of very intuitive ways of looking at color spaces. One of them is the fact that the RGB color space is viewed like a unit cube made up of color with the main diagonal from the origin to the point opposite to the main diagonal symbolizing the grayscale representation. HSV on the other hand is viewed as a cylinder with the hue being the curvature in the cross section circle of the cylinder, the value changing as a function of the height of the cylinder as well as the saturation representing the radius from the vertical axis of the cylinder. HSV color space can be realized as a function of the more approachable RGB space as follows:

$$r, g, b = \frac{r'}{255}, \frac{g'}{255}, \frac{b'}{255}$$
$$M = \max(r, g, b)$$
$$m = \min(r, g, b)$$
$$c = M - m$$
$$s = (\frac{c}{M})100$$
$$R, G, B = \frac{M - r}{c}, \frac{M - g}{c}, \frac{M - b}{c}$$
$$H = 60°(\frac{G' - B'}{\Delta} \mod 6)$$

One of the more major techniques we studied was white balancing. The technique forms the backbone of many video and image editing software and is clearly relevant for our application where the illumination of scenes with varying Jerlov types has blue or green tints that we want to eliminate. Before talking about the technique used to white balance, it is important to understand what exactly white balancing is. White balancing aims to balance the color temperature of an image. Now what is the color temperature? Color temperature is a physical property of the illumination source. It is measured in Kelvins and can be defined in terms of black body radiation. Specifically, it is the temperature at which a black body radiates the color similar to the color radiated by a light source. Color temperature is a characteristic of visible light that has important applications in lighting, astrophysics, horticulture, and other fields. One very straightforward example is when watching different incandescent light bulbs in a row. Although they might all have been made by the same manufacturer, the light color they radiate varies quite a bit. We can also look at the biggest light source we have- the sun! At different times of the day, the sun radiates light with colors that vary.

In our problem, we have a color temperature that is heavily biased towards blue and green and this is due to the fact that the red color wavelength is attenuated more than the other wavelengths. This property is called selective attenuation of light. Therefore there is a need to white balance this image and remove the blue tint. As the name suggests, white balance balances the color temperature in your image. How does it do this? It adds the opposite color to the image in an attempt to bring the color temperature back to neutral. Instead of whites appearing blue or orange, they should appear white after correctly white balancing an image.

We implement a simple hue shift algorithm that averages the RGB channel information to create a channelwise gain after comparing it to the average of each channel separately. Then it performs a weighted addition of the different channels to create a hue corrected image. As we will see shortly, this very simple algorithm performs better than many DL based methods! We study more standard white balancing techniques: white patch based balancing, gray world assumption and an integrated openCV based learnable model with transfer learning on our dataset.

The white patch algorithm is an algorithm that searches for the lightest patch to use as a white reference similar to how the human visual system does. Remember when humans go out into broad daylight after staying in a dark room for some time? This is exactly the procedure the human eye follows! It also looks for a reference patch and then normalizes the vision. More generally, The objective of this algorithm is to normalize each image channel to the specific value from that channel. This can be done in various ways. The most common way is to threshold to the maximum. However this is not always successful. More commonly, people try to use mean values as a replacement for the maximum. In fact, we implement a percentile based system so that we can evaluate at different specific values: the mean, the median, the max and so on and so forth. Specifically, the ninety ninth percentile works very well in our specific use case and we find this through hyperparameter tuning.

A more common assumption in white balancing is to assume that the image in reference is composed of a good distribution of colors. This will therefore imply that the average color in the image is gray. This is exactly the assumption referred to as the gray world assumption. The gray world algorithm operates on this assumption and calculates the average reflected color in the image and assumes it to be the color of the light source itself. Using this information, we can estimate the illumination of a scene by looking at the average color and comparing it to the average which is gray. This algorithm is also quite simple to implement and is not computationally intensive. However it is important to note the fact that our images are not like this! Due to selective attenuation of light, our images have very little red and therefore the gray world assumption does not hold in principle. Due to the high blue-green tint with no plausible red in images with high Jerlov numbers, this algorithm falters.

We also try a learnable opencv white balancing

class object with saturation 0.99. The default model is used which is trained on Gehler-Shi dataset and performs very satisfactorily. This work uses 2D features with multiple regression trees. The method avoids the use of standard signal processing techniques widely used in image processing such as spectral transforms, moment based methods and even convolution. This regression tree based model is therefore both computationally efficient and easy to apply. The training of the trees is also comparatively fast when compared to standard deep learning techniques. As a benchmark tool, we also use some hue correction tools provided in IMovie and notice that they perform very well while being real time.

4.2 Dark Channel Prior

Before DCP, much work had been done on the atmospheric light based model that DCP adopts. Particularly there was work by Tan et al. that maximized the visibility (sum of gradients) of patches in the image with an inequality constraint on the scene radiance. They further adopt the usage of a Markovian Random Field based method that regularizes the images. Another method was by Fattal et al which adopted the usage of ICA (Independent Component Analysis) to estimate the albedo of a local patch. These methods tend to have saturation and discontinuity based issues and artifacts. Also some independence assumptions they make are not really justified.

$$I(x) = J(x)t(x) + (1 - t(x))A$$

This motivated the need for a more comprehensive technique using the atmospheric light model postulated in the early 2000's. Developed by Kaiming He's team in 2011, the Dark Channel Prior based method is based on the following observation on haze-free outdoor images: in most of the non-sky patches, at least one color channel has very low intensity at some pixels. In other words, the minimum intensity in such a patch should have a very low value.

This assumption is not unfounded. In Fact for many outdoor scenes, this assumption is rather intuitive. One of the easiest explanations is the presence of black objects in the scene. However even without black objects, shadows of different objects as well as vivid colors lead to this assumption/observation being verified. The dark channel is therefore valuable information we can use to create a prior. The authors formulate this problem as a double minimisation. First they solve for the minimum of each pixel across the different channels to extract the value that has the lowest intensity in one of the three channels. Further, the authors perform a minimum filter with patches of size 16X16 to create a minimum of the minimum. This minimization problem when integrated with the equation for atmospheric light leads to a pixel wise minimisation problem which when combined with the soft matting techniques for incorporating spatial continuities give a dehazed image.

In 2010, Chao et al. used the DCP to recover the underwater images and remove the scattering of underwater images, respectively. This is also something we tried and it was very noticeable that the DCP method did not perform well with it also adding to the hue shift and discoloration effects. Thus DCP is not directly transferable. We postulate this is because of the selective light attenuation problem discussed before. In 2011, Yang et al. proposed a DCP-based method that replaced soft matting with median filtering and finally used a color correction algorithm for chromaticity correction. However this method also suffers from some issues. In high jerlov water types, the dynamic range in terms of the color is low and their method fails.

4.3 Underwater Light Attenuation Prior (ULAP)

Song et al. proposed a fast, real-time underwater physics based model for color correction and scene depth estimation. This is especially relevant for our application in underwater SLAM where we want to detect the trajectory of the AV drone. They proposed the method called Underwater Light Attenuation Prior also referred to as ULAP. This method plays around with RGB intensities, specifically the difference between the maximum value of G-B intensity and the value of R intensity in one pixel of the underwater image and postulates that this is highly correlated to the change in scene depth. Based on the ULAP, a linear model was established to rapidly obtain scene depth maps, which can be used to estimate the background light (BL) and transmission maps (TMs) for R-G-B channels are easily estimated to recover the true scene radiance under the water.

The background light BL in the atmospheric light equation is often estimated as the brightest pixel in an underwater image. However, the assumption is not correct in some situations, e.g., the foreground objects are brighter than the background light. The background light is selected from the farthest point of the input underwater image, i.eThe position of the maximum value in the refined depth map corresponding to the input underwater image is the background light candidate value. But directly selecting the farthest point as the final background light, some suspended particles can interrupt the valid estimation result. After generating an accurate depth map, we firstly remove the effects of suspended particles via selecting the 0.1% farthest point, and then select the pixel with the highest intensity in the original underwater image. For depth maps, this method uses manually modeled priors and considers factors such as Pearson correlation coefficients to find the correct prior for the test case. This is also a variant of DCP often referred to as the Bright Channel prior method and works very well for underwater test cases.

4.4 Underwater Haze Lines

The third and final rigorous model we study is the underwater hue correction model developed with the use of Haze Lines, a technique which is an extension of the DCP based method proposed by Kaiming He and his team. The authors postulate that " Under water, where the assumption of wavelength-independent attenuation does not hold, there are theoretically three unknown transmission values per pixel, one per channel, yielding six unknowns with only three measurements. However, the color-dependent transmission is related to the distance via the attenuation coefficients. Based on this relation, the problem is reduced to estimation of four unknowns per pixel as before, with two new global parameters, the ratios between the attenuation coefficients of the color channels."

This problem can be solved if we have multiple different images of the same scene but often the creation of this prior is of concern since these images can not always be created. This is the same argument that the DCP paper also follows. The degradation depends both on the water properties and the 3D structure and can be estimated by analyzing multiple images that satisfy certain conditions using orthogonal polarizer angles and utilizing the partial polarization of light to restore the visibility. These viewpoints can help constrain the complexity of the problem but as said before this constraint is often not satisfied.

The authors use the Haze Line technique by making the observation that clustered pixels are usually non local and are located at different spatial coordinates with respect to the optical sensor. Since these clusters translate to different transmission coefficients or in our case transmission coefficient ratios, each cluster becomes a line in RGB space. Therefore with the knowledge of the global attenuation or transmission ratios the problem reduces to a single image dehazing which has many efficient algorithms to be solved with. The problem now lies in the estimation of these global parameter ratios. In this specific approach by the authors, different Jerlov Water types are studied and then the best one corresponding to the scene at hand is chosen through statistical methods such as correlation coefficients. Once the prior is known, assumptions such as Gray World Assumptions can be made. We said before that the GWA is a bad assumption for our problem. However it is important to notice that after dehazing, the GWA is okay to use since the dehazing improves the color dynamic range leading to the assumption being fulfilled.

Now all that's left is to understand how exactly this scene radiance estimation takes place. We need to estimate the veiling-light, which is required in every dehazing algorithm. The assumption is that a BCP type system will be fulfilled in areas where there is no object. Therefore an area where the object is not present can be used as long as the region is smooth and texture less. This can be done using standard edge map techniques. The authors use Structured Edge Detection Toolbox with pre-trained model and default parameters. The edge map is thresholded and the largest component is selected with the pixels that belong to this area being the veiling light estimators. A simple average color inside this region leads to the veiling light being estimated.

Algorithm 1: Underwater Haze Lines

Data: $I(x)$
Result: $t(x), J(x)$
Detect veiling light pixels and calculate A;
for *each* (β_{BR}, β_{BG}) *values of water types* **do**
  for *each* $c \in (R, G, B)$ **do**
    $Ic(x) = sign(Ic(x)Ac)abs(Ic(x)Ac)^{\beta_{Bc}}$
  end
  Cluster pixels to Haze-Lines and estimate transmission
  Apply soft matting to t_B *with the lower bound*
  Regularization using guided image filter, with a contrast-enhanced input as guidance
  Calculate the restored image
  Perform a global WB on the restored image
end

5 Deep Learning approaches

Using learning-based methods on images to remove the degradation caused by the presence of water is a difficult task for multiple reasons. Lack of ground-truth data presents a major obstacle for under water image color correction. It is difficult to train a supervised learning-based method when there is a lack of available datasets with a ground truth to teach a network how an underwater image should look like outside of water. Likewise, obtaining a dataset that contains a scene, then submerging the scene in water, and recapturing the exact same scene is also inherently difficult as it requires lots of resources. The only way to generate enough paired data to sufficiently train a learning-based method is by augmenting data that was captured above water. A popular augmentation technique is to perform style transfer on the in-air data and add distortion to make it appear as though it were under water. This results in a pair of images: the original in-air image for ground-truth and the augmented data for the distorted image.

Another prominent issue is simply the large diversity of water types that can be captured in images. The appearance of images captured underwater have lots of variance due to factors such as light attenuation and scattering, which can cause undesirable effects in the image. Consequently, this makes the image difficult to use for other tasks. The dataset used to

train a learning-based method that is generalizable to the wide distribution of underwater images must account for all water types of the Jerlov classification scheme. The categories are determined by the inherent optical properties of water, from which, a metric known as the diffuse attenuation coefficient arose. The diffusion attenuation coefficient is a measure of how light dissipates with depth in water. A higher diffusion attenuation coefficient corresponds to a lot of attenuation as light passes through, and a lower diffusion attenuation coefficient corresponds to more transparent-appearing water. The Jerlov classification scheme distributes water types between 5 categories of Oceanic waters and 9 categories for Coastal waters based off the water's diffusion attenuation coefficient.

Ideally, the best model to use will have been one that was both trained on a dataset that accounts for all of the Jerlov water types and has access to ground truth data. To our best knowledge, there is no dataset that has real ground truth for the task of underwater image color correction, so we sought models that synthetically generated the various Jerlov water types.

5.1 Color Correction Architectures

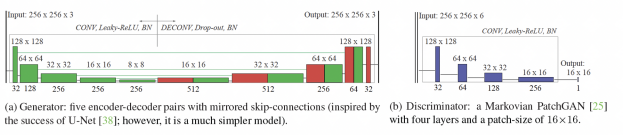
We tried 3 deep learning techniques for underwater image color correction.

5.1.1 FUnIE-GAN

The first model that we tried, FUnIE-GAN (Fast Underwater Image Enhancement), is a conditional GAN-based model that learns the mapping between a source domain composed of images that appear to be taken underwater and a target domain of the same images but with the water removed. FUnIE-GAN was inspired by U-Net which is an encoder-decoder model with skip connections, but it was implemented with far fewer parameters to improve speed of inferences for the application of underwater image color correction in real time. The FUnIE-GAN model was trained on the EUVP dataset which consists of 12k paired images and 8k unpaired images captured from 7 different types of cameras.

5.1.2 UIE-DAL

The second model, UIE-DAL, uses an Encoder-Decoder to learn the mapping between source and target domain. It also has a nuisance classifier that attempts to identify the Jerlov Water type of the encoder output. The encoder uses the nuisance classifier as input for its adversarial loss function. The adversarial loss makes it so that the encoder generates an encoding that is water type agnostic meaning the nuisance classifier should not be able to distinguish the water type.



(a) Generator: five encoder-decoder pairs with mirrored skip-connections (inspired by the success of U-Net [38]; however, it is a much simpler model).

(b) Discriminator: a Markovian PatchGAN [25] with four layers and a patch-size of 16×16.

Figure 2: FUnIE-GAN Architecture

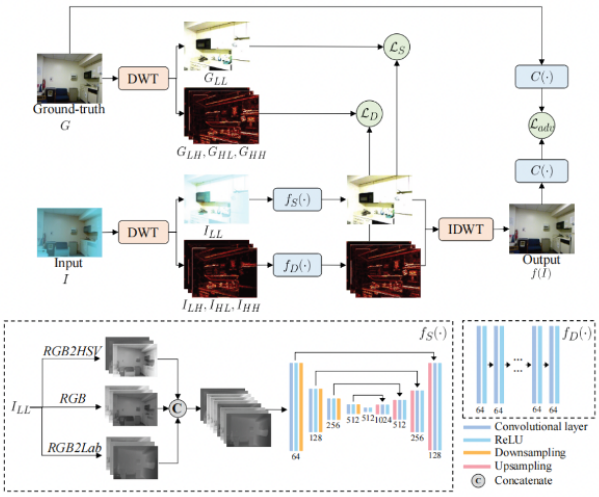


Figure 3: Wavelet GAN Architecture

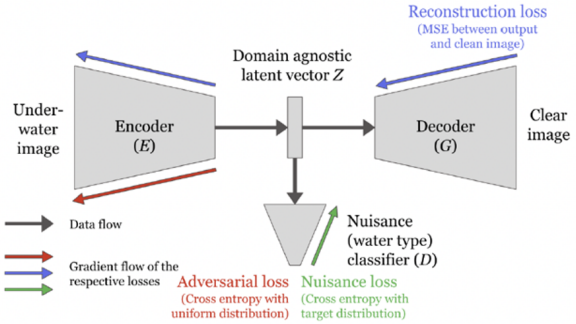


Figure 4: UIE-DAL Architecture

5.1.3 Wavelet-Based Dual-Stream Network

The last model that we tried was a wavelet-based dual-stream network. This model utilizes the discrete wavelet transform. Just a quick overview of the DWT. The 2D Discrete Wavelet Transform is a technique used to capture both the spatial and frequency content of an image. It performs a recursive decomposition of the image, by splitting the image into quadrants. So the lower left and lower right quadrant, LH1 and HH1 respectively, correspond to lower frequency details of an image, the upper 2 quadrants correspond to the higher frequency details. This is a recursive algorithm because the upper left quadrant is further decomposed into another 4 quadrants where LH2 and HH2 correspond to the lower frequencies of the original higher frequency range determined by the first decomposition. An n-level DWT computes this process of decomposing the upper left quadrant n times. After performing an n-level DWT, a perfect

reconstruction of the original image can be attained by performing an inverse Discrete wavelet transform using the coefficients from each quadrant at each level, so no information is lost when performing the Discrete Wavelet Transform.

So back to the model. This was the model that gave us the best results. The model consisted of a multi-color space fusion sub-network and a detail enhancement sub-network. The multi-color space fusion network utilized U-Net as the base architecture, but modified the original RGB input to be a 9-channel input where the first 3 channels correspond to the original RGB channels, the next 3 correspond to the RGB channels transformed to HSV color spaces, and the last 3 correspond to the RGB channels transformed to Lab color spaces. The goal of the concatenation of the 3 different color spaces into a single input was to further extract diverse feature representations of a given input without adjusting the architecture of the network to be deeper. Whereas the multi-color space fusion sub-network was designed to improve color, the detail enhancement network was designed to improve the detail of the image. During training, the network tries to learn from the coefficients that correspond to the Discrete Wavelet Transform of the ground truth data so that it can reconstruct the training image without it appearing to be captured by using the inverse discrete Wavelet Transform. Then a GAN is used for adversarial learning, and is trained to distinguish the ground truth G from the output of this network.

5.2 Brief Comparison

The UIE-DAL and wavelet-based dual-stream network were actually trained using two of the same datasets: the NYU-v2 dataset which consists of 3000 images and the UIEB dataset which contained 890 images. Unpaired images are images that were captured underwater that have no ground truth. Even though there is no ground truth, these images are useful in training because it incorporates real world data into the training. These images are used to enforce consistency in the synthetic generation of training data. The authors of the FUnIE-GAN paper mention that their "focus is to facilitate perceptual image enhancement for boosting robotic scene understanding, not to model the underwater optical degradation process for image restoration." Observing this statement and their choice to utilize a simplified network, the results we find make sense.

6 Different SLAM utilisations

6.1 Bayesian model

Estimation methods utilizes prediction and correction of the state via an estimator. Which in SLAM is to predict the motion relative to the camera in the prior frame and then do map correction between the estimated state and the actual state. Therefore, we can model the map as the desired state which includes the robot state and the Landmark states.

$$x = \begin{bmatrix} R \\ L_1 \\ L_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} R \\ M \end{bmatrix} \tag{10}$$

When the robot moves, we know that the relative position of the landmarks and the robot itself would also be affected by the movement of the robot. Therefore, if we have the actual movement of the robot, we could then find the change in relative position by a function

$$\dot{L}_n = f_n(L_n, \dot{x}) \tag{11}$$

The issue is when we don't have a reliable way to take the actual movement of the robot accurately like if we only have a monocular camera setup on board. Implying that we need something else that we know would yield a reliable change of position that could be measured, which so happens are the landmarks themselves. Thus, since we do predictions and updates on a frame-by-frame basis, we can model the system as a discrete nonlinear system with noise

$$x_k = f(x_{k-1}) + W_{k-1}, y_k = gx_k + V_k \tag{12}$$

Where W_{k-1} is the state noise and V_k is the output noise, but since the point of SLAM is to get the estimation of the local map and the robot position the output noise could be ignored without loss of generality. If we then assume that both W_{k-1} and V_k are Gaussian random variables we could utilize a Bayesian filter to attempt to filter out the noise. Which is modeled as a random process

$$p(x_k|Z_k, U_k)p(Z_k|x_k) \int p(x_k|x_{k-1}, U_k)p(Z_{k-1}, U_{k-1})dx_{k-1} \tag{13}$$

U_k being the known characteristics for which the robot is driven from state x_{k-1} to x_k, and Z_k being the observations made from the sensory inputs at time k. Therefore the integral part of the equation is basically a prediction of the next state, which is mathematically the marginal distribution of the predicted state from the prior state. Then after correcting the prediction by applying the updated state of the robot R_k to the position of the landmarks in the state M_{k-1} and verifying the results with the observations Z_k we could get an updated posterior at time k.

Two prominent approaches of the Bayes filter on SLAM are the Particle filter or Extended Kalman filter (EKF). Therefore we chose ORBSLAM3, which implements the EKF and DSO, which implements the particle filter. The full derivation of the methods itself is out of the scope of this project.

6.2 Extended Kalman Filter

For LTVC(Linear Time Varying Causal) systems (since the system is time varying state variables A, B, C should be $A(t), B(t), C(t)$ but I'll abuse the notation and cut out the (t) for convenience) the Kalman filter is the optimal linear estimator, which is a full state estimator when the system is observable

$$\dot{x} = Ax + Bu + W, y = Cx + V \tag{14}$$

We define that $\hat{x}$ is the estimated state from the output of the system, we can derive a model from applying a gain l to the difference between the actual output of the system and the estimated system.

$$\dot{\hat{x}} = A\hat{x} + Bu + W + l(y - C\hat{x}) \tag{15}$$
$$= (A - lC)\hat{x} + Bu + W + ly \tag{16}$$

Different from a standard closed-loop state estimator is that since we now have $ly = lCx + lV$ the gain has to be chosen carefully because in this case we can see that the output noise would be proportionally more impactful on the estimated system with respect to l. Applying a change of variables $\epsilon = x - \hat{x}$, we could get

$$\dot{\epsilon} = (A - lC)\epsilon - lV \tag{17}$$

We can view ϵ as the error of the estimator with respect to the real state. Ideally, we can choose a gain l s.t. the system is asymptotically stable in the sense of Lyapunov, $\epsilon \to 0$ as $t \to \infty$. If there is no noise in the system this problem would be quite trivial since we can easily choose l s.t. the eigenvalues of $(A - lC)$ have negative real parts and converge exponentially quick to the equilibrium 0, but in the case of the system being noisy we would have to optimize between the speed of error convergence and the impact of the noise. Which is what the Kalman filter is designed for by utilizing a covariance estimate to find an optimal Kalman gain.

6.3 Particle Filter

In the case of the particle filter in SLAM we assume that the state of the system is a Markov Chain X_t, and the observation with noise a random process Y_t, the initial distribution of the initial state is 0. Then the conditional probability of $X_t = x_t$ given $X_{t-1} = x_{t-1}$ is $P(dx_t|x_{t-1})$, and the conditional probability of $Y_t = y_t$ given $X_t = x_t$ is $g(y_t|x_t)$. In other words, P is the distribution of the state estimation from the last state to the current state and g is the distribution of observing the system. The particle filter is a Monte Carlo approach to the problem. Therefore, it utilizes random sampling to create "particles" and recursively computes importance sampling approximations which is

$$\pi_n(y_{1:n}) = \sum_{i=1}^{N} W_n^i \Delta_{X_n^i} dx_n \tag{18}$$

Where π_n is the importance sampling approximation, N is the number of particles implemented within the system, x_n is the current state, y_n is the observed system, X_n^i are the particles which are random variables. Therefore, we can see that the method is a weighted sum of particles which is essentially a variation of the marginal distribution of the estimation of the next state from the previous observed states. The intuition behind could be generally though of as each particle is what a robot deems to be a possible state, then through updates of observations the particles get resampled proportional to their weights.

$$W_n^i = \frac{target(X_n^i)}{proposal(X_n^i)} \tag{19}$$

Therefore, since particle filters assign weights to different possibilities of the state, the biggest advantage of a particle filter over a EKF is the handling of nonlinearities and non-Gaussian noise due to the assumption that the noise is a random process in general. The downside of the particle filter is that the more particles you use the computational complexity would be an issue since the estimation and correction would need to be updated in every single particle instead of treated as a closed loop system in general.

7 Qualitative Comparison

When evaluating the performance of in air SLAM a common practice is to compare the output of the SLAM with the ground truth and evaluate the error between. However the cost and complexity of obtaining a ground truth underwater is huge. Currently the most common way of doing this is the Short Baseline Positioning Method. Which is the term for using a surface robot mounted with sonar to ping the underwater robot. Meaning that we would need another robot as well as a sonar with decent accuracy and refresh rate at the range of 15 meters. Where the last time we checked a sonar that can do this but yields varying results alone would cost around 40 thousand dollars, let alone the cost of the surface robot and the operation cost. Funnily enough the most consistent SLAM in this area would be our own eyes since we have built-in biological SLAM mechanisms ourselves. Therefore we would have to come up with creative ways to evaluate the results of SLAM.

7.1 SLAM results

The output of SLAM is a file containing the keyframes detected within the whole trajectory. In both ORB-SLAM3 and DSO they are both outputted in the format

timestamp x y z q_x q_y q_z q_w where q is the orientation of the robot in quaternion, in our case since our focus is on the trajectory of the robot the orientation is not included in our calculations.

using the timestamp and x y z we plot out the trajectories using matplotlib such as the ones in Figure 1

7.2 Human estimation

This is rather straight-forward. You watch the underwater video and come up with what the trajectory should look like and make a critical analysis based on that. Which is the method used in Figure 1 to determine that FUnIE-GAN performs better than the unprocessed images on the synthetic dataset.

7.3 Keyframe analysis

Another approach we tried to utilize is the number of keyframes being detected from SLAM. SLAM keyframes is the term used to describe a frame that yields a high confidence regarding an update to the local map. Which means that keyframes are typically frames that either have a lot of features or has a high ratio of overlapping features with respect to the last keyframe. Therefore we hypothesise that the keyframe to total frame ratio would give us some insight on how the image processing method is performing since theoretically more keyframes detected would mean either more robust features are being found or features are less affected by noise and are being detected more consistently.

7.4 Color Correction modules

7.4.1 Synthetic Dataset

Our comprehensive experimental results are in Appendix A

- Underwater Haze Line - This traditional image processing technique fairs pretty well. It removes a lot of the water appearance in the foreground effectively. It is also really effective in restoring color to the petals and bushes and brightening up the video substantially. In the restored image, one is able to make out features points in the distance. Some flickering and changes in brightness occurs as we progress through the video. Relevant figures are in the appendix here.

- Histogram equalization - This appears to be a very consistent technique. There is hardly any flickering. For the task of hue correction, it is interesting to note the colors of the feature points do not appear to be the true colors of the objects. The rocks close in the foreground appear to be oversaturated. However, compared to the other techniques, this appears to generate the most coherent video for the usage of SLAM as the appearance of water appears to be completely removed and the full shape of objects are easy to distinguish as feature points for the SLAM algorithms. Relevant figures are in the appendix here.

- HSV with Equalization - This technique is able to generate a video that is on par with that of normal histogram equalization. It maintains all of the same advantages but removes the appearance of oversaturation in the rocks. The result is essentially a gray scale video with the same clarity of the video generated by performing histogram equalization.

- White balancing and Hue correction - Although it wasn't as effective at removing the appearance of water as the other techniques, it does still remove a significant percentage of the appearance of water. The background is not as easily distinguishable as the previously mentioned techniques though. This technique also introduces some very ostensible flickering. Relevant figures are in the appendix here and here.

- White Balancing w/ 99 Percentile – This is an improvement over WB Hue correction in that the flickering is removed and the objects in the video appear to have more natural colors. Also, because the brightness is increased, it is also easier to view the objects in the background. Relevant figures are in the appendix here.

- FUnIE-GAN – This technique removes the appearance of water, but the hue correction makes it appear as though it were captured on another planet like Mars or in the desert. There are strange artifacts towards the top of the video and significant flickering is present and the background is very blurry. Relevant figures are in the appendix here.

- UIE-DAL – Out of all the techniques, the flickering was the worst with this one. Although it is effective in removing color, the random and incoherent flickering is so disturbing that it becomes difficult to observe the objects in the video. It almost appears as though there are several illumination sources flickering on and on rapidly. It is highly distracting and has the potential to significantly impact the performance of SLAM since SLAM depends on the feature points from frame to frame. Relevant figures are in the appendix here.

- Wavelet GAN – This technique is troubled by some random illumination like in UIE-DAL, but not nearly as troubling. There is much more consistency from frame to frame. The color also appears to be fully restored. The rocks, grass and leaves all have a natural color to them. The background, although a bit blurry, appears to be easier to view. Relevant figures are in the appendix here.

7.4.2 Real Dataset

- Underwater Haze Lines – This technique appears to significantly remove the presence of water, but like in some of the previous techniques, it appears as though there an illumination source is above flickering on and off. Relevant figures are in the appendix here.

- DCP – The video generated by applying the dark channel prior appears to perform decently well,

but then certain frames get distorted by noise and the overall look appears to be oversaturated. Surprisingly, there is an effect in IMOVIE that appears to perfectly remove the hue from the water, produce a consistent video all the way through, and restore some of the color in the buckets beneath the submarine. Relevant figures are in the appendix here.

- White balancing and Hue correction – Although it does not perform as well as the IMOVIE effect, this also performs a good job of removing the appearance of water in the video.
- GANs FUnIE-GAN produced the same artifacts as was seen in the color correction applied to the Blender Generated Videos. The wavelet based dual stream network produced a video that did not have enough contrast to be able to identify the feature points in each frame.

Out of all the image processing techniques, Haze Line appears to produce the best video for the task of hue correction as it generates the most natural looking colors in the objects present in the videos. However, as an input to the SLAM algorithm, the histogram equalization and HSV with equalization appear to perform best at removing the presence of water in the image. Out of all of the GAN videos, the wavelet based dual stream network is by the far the best produced. Although FUnIE-GAN produced a video that did not appear to be underwater, the quality of the video was not very good and the color seemed unnatural. UIE-DAL produced far too many artifacts to be useful as input to SLAM. Overall, it appears as though the image processing techniques performed much better for the task of restoring a video for better performance with SLAM.

7.5 SLAM inference & analysis

Since we know that the feature detection algorithm implemented by ORBSLAM is ORB and DSO defines features as pixels with high gradient, it is obvious that in underwater environments in general DSO would perform better since there are not as many distinct feature points underwater than in air. Which is also shown by the SLAM results in Figures 58, 59, 56, 57 as well as the keyframe analysis in Table 1. Especially in the keyframe analysis where DSO in general have a much higher keyframe ratio in all tests.

A really unexpected result is that although the GANs didn't perform well in the context of color correction, they yield exceptional performance when feeded through SLAM. Our inference is that the GAN networks, while doing a poor job of color correcting, actually created more distinct and realiable feature points for the SLAM systems to pick up.

Another unexpected result is that the keyframe ratio for DSO on the real data is higher that DSO on the synthetic data since theoretically the synthetic data is in a more idealized environment and have more distinct features. On the other hand for ORBSLAM the result goes as expected with keyframes detected on the synthetic data being more numerous than those detected on the real data. Resulting in the results from DSO on real data are debatably more accurate than the synthetic data.

In the end it is almost impossible to say anything that is concrete due to the lack of ground truth, but the results yielded by the analysis methods used here are mostly in line with the trajectories we see.

Our comprehensive experimental results are in the Appendix A

8 Key Conclusions

Through this project we create a setup capable of doing a comprehensive evaluation of two modules. First using WaterGAN and other synthetic tools, we create our own dataset. The first module deals with performing hue correction and here we contrast several fundamental image processing methods such as Gray World Assumption and Histogram Equalization with different color spaces with both state of the art image processing techniques built on physics based models (ULAP, Haze-Lines, DCP, BCP) and deep learning techniques dealing with style transfer and underwater domain shift problems (FUnIE-GAN, UIE-DAL) along with an integration of the two field with techniques like Wavelet GAN. We also go on to analyse different types of SLAMs both mathematically as well as experimentally which form our second module. We compare the different methodologies using DSO or ORBSLAM3 on both the Blender based synthetic dataset as well as the NBRF real-world dataset setups. We come across some interesting results discussed above such as the fact that even though image processing methods perform really well *visually*, some GANs follow the *ground truth* trajectories better. Note the fact the ground truth here has a different meaning as in above ground applications. Future work may involve breaking down different aspects of underwater navigation problem using the best performing methods found in this work.

References

[1] Miao Yang, Jintong Hu, Chongyi Li, Gustavo Rohde, Yixiang Du, and Ke Hu. An in-depth survey of underwater image enhancement and restoration. IEEE Access, 7:123638–123657, 2019.

[2] Carlos Campos, Richard Elvira, Juan J. G omez Rodr ıguez, Jos e M. M. Montiel, and Juan D. Tard os. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. IEEE Transactions on Robotics, 37(6):1874–1890, 2021.

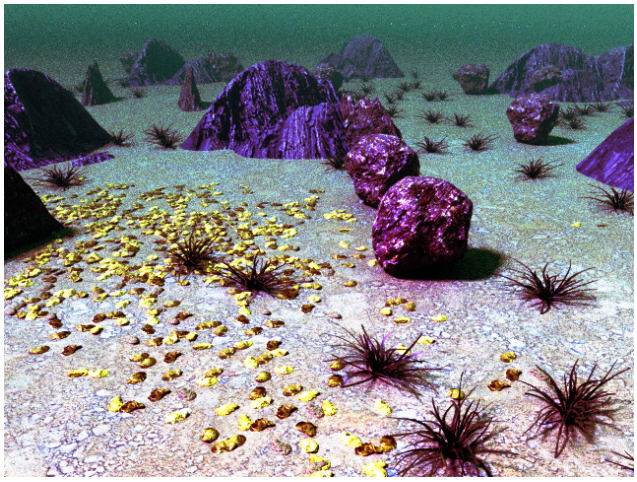


Figure 5: Channel Wise Histogram Equalization On Synthetic Dataset

Figure 6: Histogram Equalization On V channel of HSV space on Synthetic Dataset

Figure 7: Histogram Equalization On Y channel of YCbCr space on Synthetic Dataset

[3] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In arXiv:1607.02565, July 2016.

[4] J S Jaffe. Computer modeling and the design of optimal underwater imaging systems. IEEE Journal of Oceanic Engineering, 15:101–111, 1990.

[5] Jie Li, Katherine A Skinner, Ryan M Eustice, and Matthew Johnson- Roberson. Watergan: Unsupervised generative network to enable real-time color correction of monocular underwater images.

[6] B L McGlamery. A computer model for underwater camera systems. Volume 0208, 3 198

[7] Berman, D., Avidan, S. (2017). Diving into Haze-Lines : Color Restoration of Underwater Images.

[8] Kaiming He, Jian Sun and Xiaoou Tang, "Single image haze removal using dark channel prior," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1956-1963, doi: 10.1109/CVPR.2009.5206515.

[9] Song, W., Wang, Y., Huang, D., Tjondronegoro, D. (2018). A Rapid Scene Depth Estimation Model Based on Underwater Light Attenuation Prior for Underwater Image Restoration. R. Hong, W.-H. Cheng, T. Yamasaki, M. Wang, C.-W. Ngo (.), Advances in Multimedia Information Processing – PCM 2018 (. 678–688). Cham: Springer International Publishing.

[10] M. J. Islam, Y. Xia and J. Sattar, "Fast Underwater Image Enhancement for Improved Visual Perception," in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3227-3234, April 2020, doi: 10.1109/LRA.2020.2974710.

[11] Ma, Z., Oh, C. (2022). A Wavelet-based Dual-stream Network for Underwater Image Enhancement. ArXiv:2202.08758 [Cs, Eess]. http://arxiv.org/abs/2202.08758

[12] Uplavikar, P., Wu, Z., Wang, Z. (2019). All-in-one underwater image enhancement using domain-adversarial learning. ArXiv:1905.13342 [Cs]. http://arxiv.org/abs/1905.13342

[13] Zadeh, L. A. (1973). Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. and Cybernetics IEEE Transactions on Systems, Man, SMC-3(1), 28–44. doi:10.1109/TSMC.1973.5408575

[14] Messner, B., Tilbury, D. (2011). Control Tutorials for MATLAB and Simulink.

[15] Fitzgerald, A. E., Kingsley, C., Kusko, A. (1975). Máquinas elétricas: conversão eletromecânica da energia, processos, dispositivos e sistemas. McGraw-Hill.

[16] Tefili, D. (2017). LAB01 – Contatos NA e NF. https://tefili.webnode.com/$_f iles$/200000039 – $edd5beecf2/LAB01$

[17] Tefili, D., Barrault, G., François, G., Ferreira, A. A., Cordioli, J. A., Lettnin, D. V. (2013). Implantes cocleares: aspectos tecnológicos e papel socioeconômico. Revista Brasileira de Engenharia Biomédica, 29(4).

9 Appendix A

9.1 Comprehensive Results

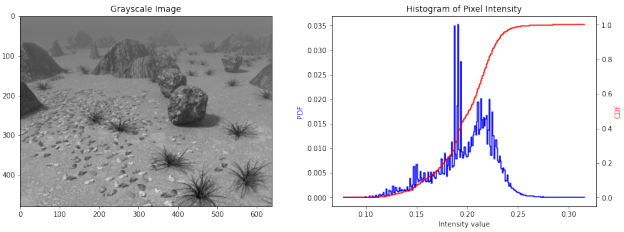


Figure 8: Grayscale approach on Synthetic Dataset

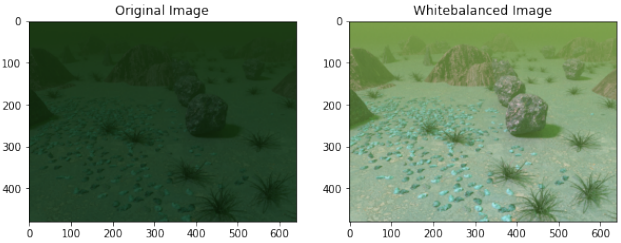


Figure 9: Maximum White Balancing on Synthetic Dataset

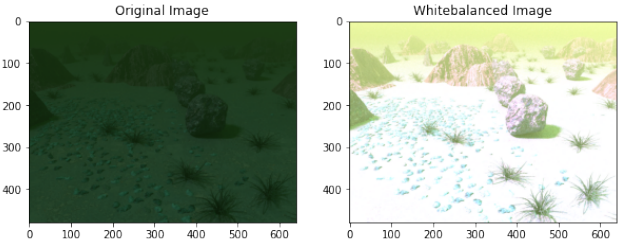


Figure 10: Mean White Balancing on Synthetic Dataset

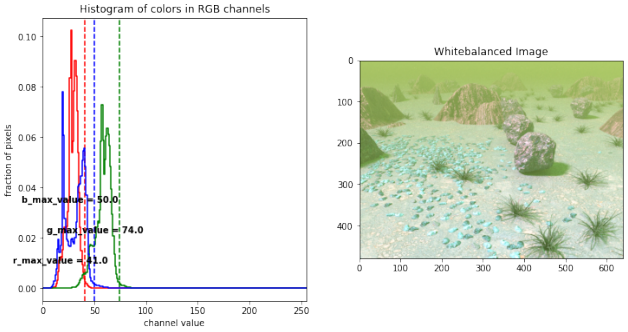


Figure 11: Percentile based White Balancing on Synthetic Dataset

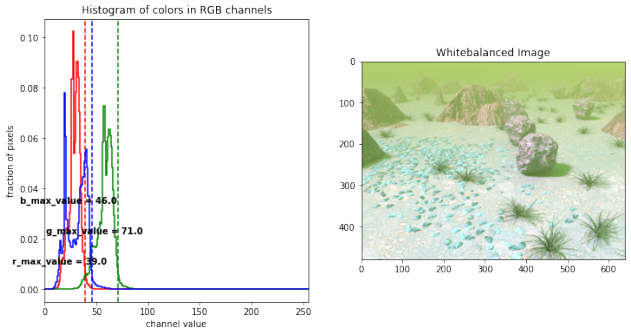


Figure 12: 97.5 Percentile based White Balancing on Synthetic Dataset

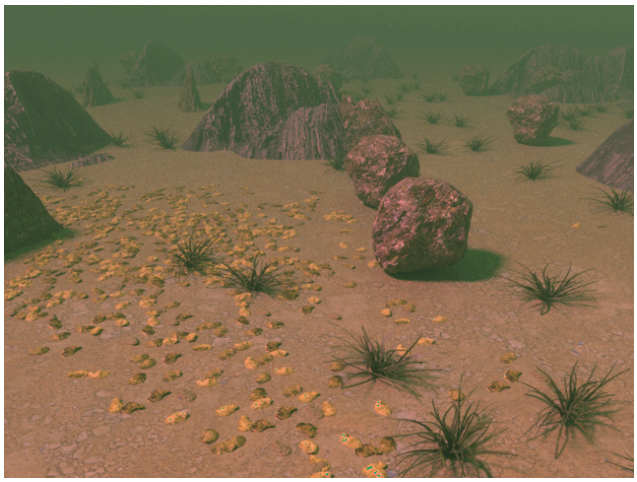


Figure 13: Channel gain based Hue correction on Synthetic Dataset

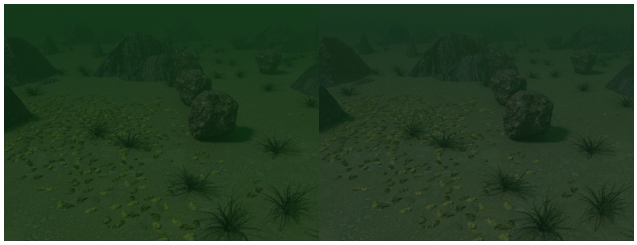


Figure 14: Fixed Gain White Balancing on Synthetic Dataset

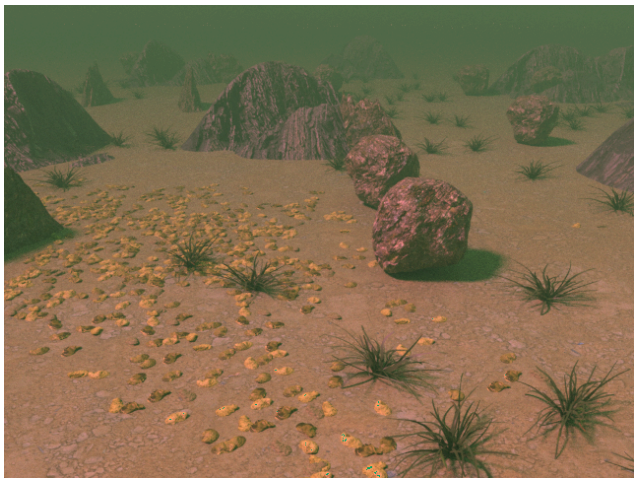


Figure 15: Adaptive gain based White Balancing on Synthetic Dataset

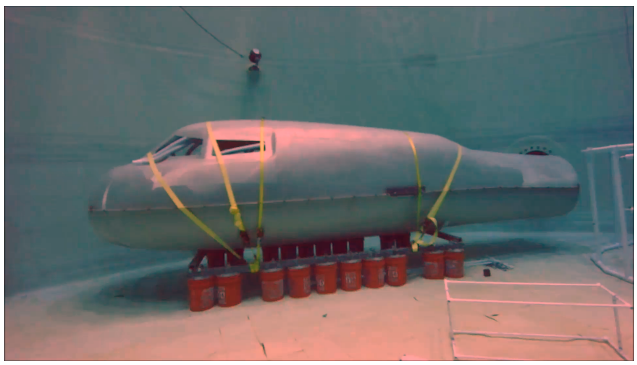


Figure 16: Adaptive gain based Hue correction on Real Dataset

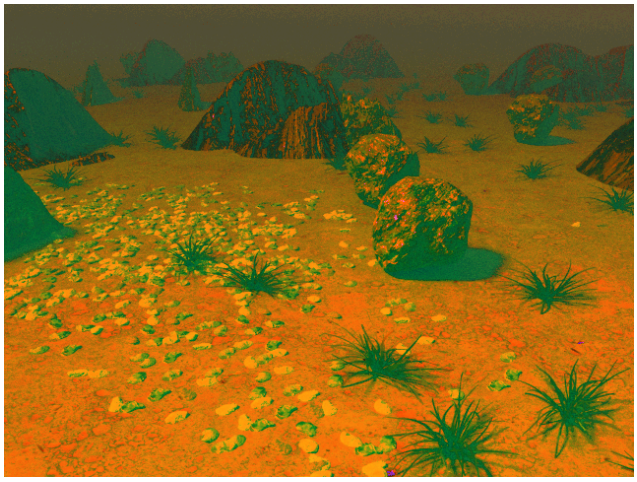


Figure 17: Playing around with HSV and HistEq

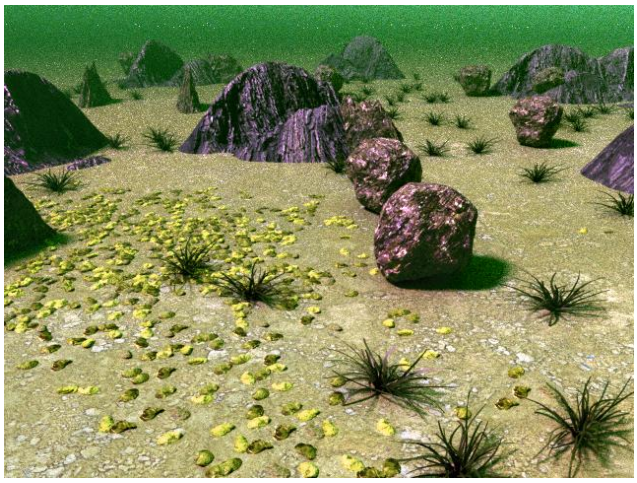


Figure 18: Underwater Haze Line method on Synthetic Dataset

Method	keyframes	frames	keyframe ratio
Haze Line	48	500	0.096
FUnIE	53	499	0.106
HistEqual	51	499	0.102
HSVnEqual	56	499	0.112
HSVwEqual	48	499	0.0962
org	52	500	0.104
Wavelet	59	499	0.118
WB	66	499	0.132

(a) ORBSLAM3 performance on synthetic data

Method	keyframes	frames	keyframe ratio
Haze Line	16	300	0.0533
DCP	41	303	0.135
FUnIE	34	837	0.0406
HistEqual	22	838	0.0263
IMovieEnhance	23	837	0.0275
org	29	838	0.0346
Wavelet	65	499	0.130
WB	0	499	0

(b) ORBSLAM3 performance on real data

Method	keyframes	frames	keyframe ratio
Haze Line	62	500	0.124
FUnIE	73	499	0.146
HistEqual	62	499	0.124
HSVnEqual	98	499	0.196
HSVwEqual	58	499	0.116
org	99	500	0.198
Wavelet	99	499	0.198
WB	81	499	0.162

(c) DSO performance on synthetic data

Method	keyframes	frames	keyframe ratio
Haze Line	43	300	0.143
DCP	69	303	0.228
FUnIE	156	837	0.186
HistEqual	171	838	0.204
IMovieEnhance	215	837	0.257
org	233	838	0.278
Wavelet	79	499	0.158
WB	118	499	0.236

(d) DSO performance on real data

Table 1: Keyframe analysis results

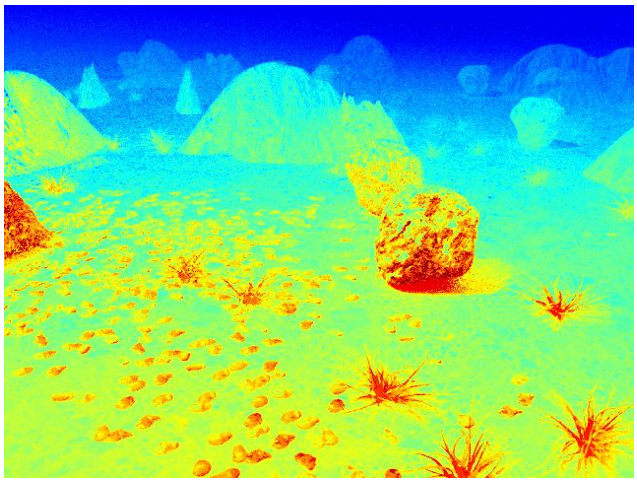


Figure 19: Underwater Haze Line method's Transmission Map on Synthetic Dataset

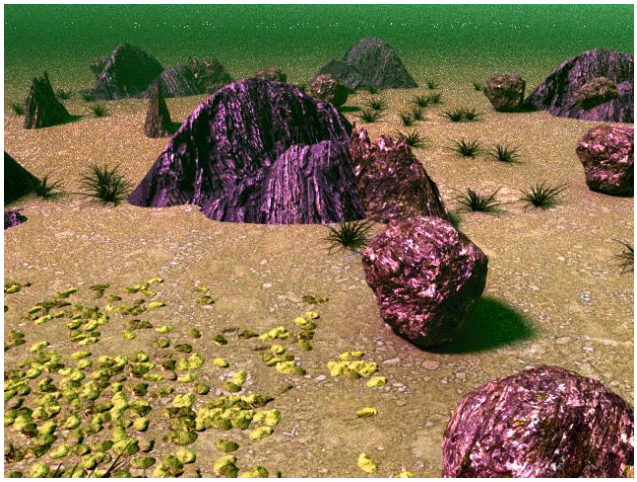


Figure 20: Underwater Haze Line method on Synthetic Dataset

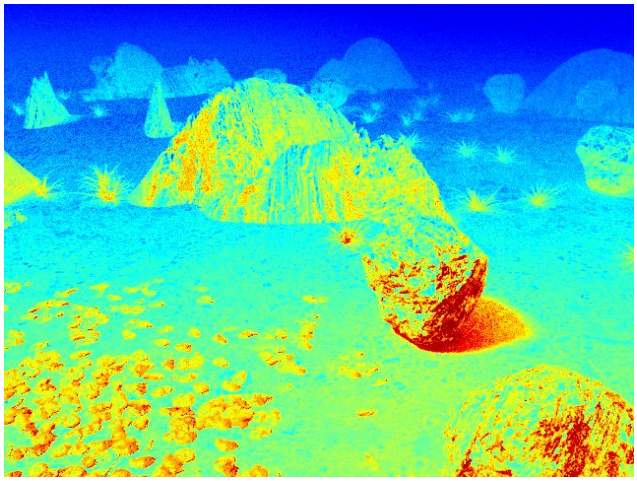


Figure 21: Underwater Haze Line method's Transmission Map on Synthetic Dataset

Figure 22: Underwater Haze Line method on Synthetic Dataset

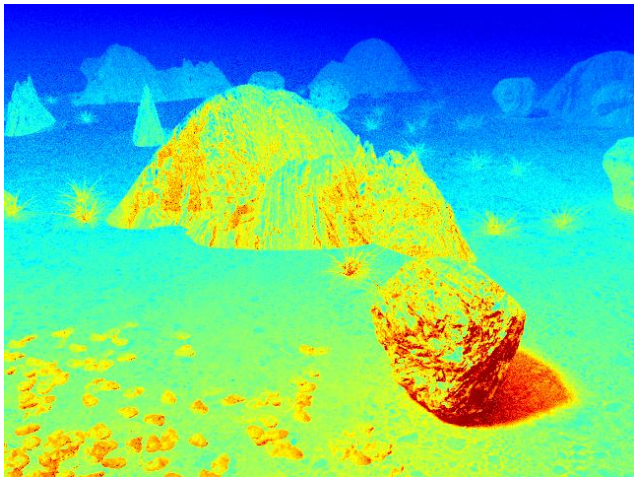


Figure 23: Underwater Haze Line method's Transmission Map on Synthetic Dataset

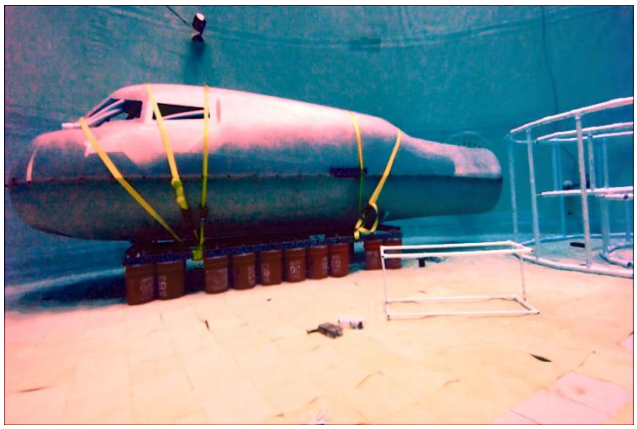


Figure 24: Underwater Haze Line method on Real Dataset

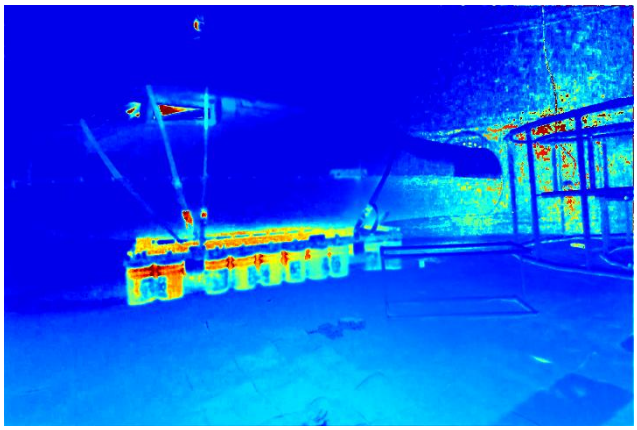


Figure 25: Underwater Haze Line method's TM on Real Dataset

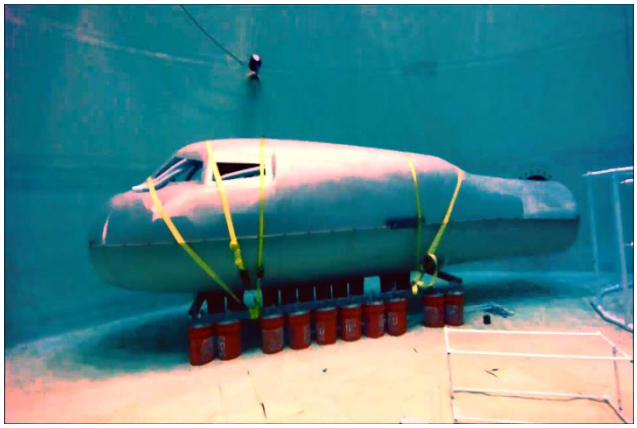


Figure 26: Underwater Haze Line method on Real Dataset

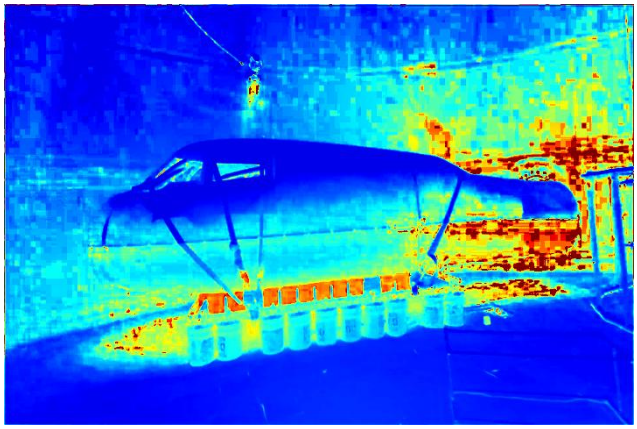


Figure 27: Underwater Haze Line method's TM on Real Dataset

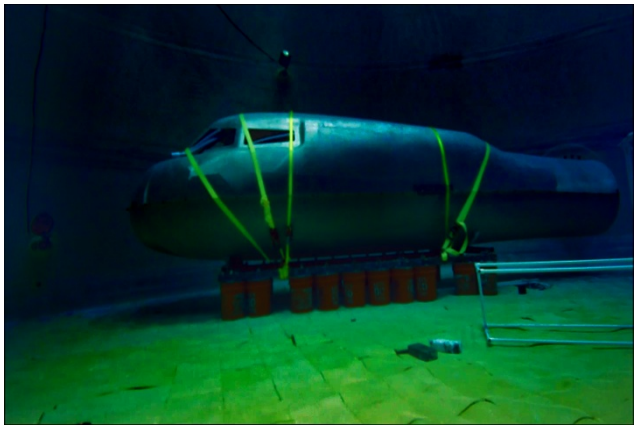


Figure 28: Underwater DCP method on Real Dataset

Figure 29: Underwater DCP method's Depth Map on Real Dataset

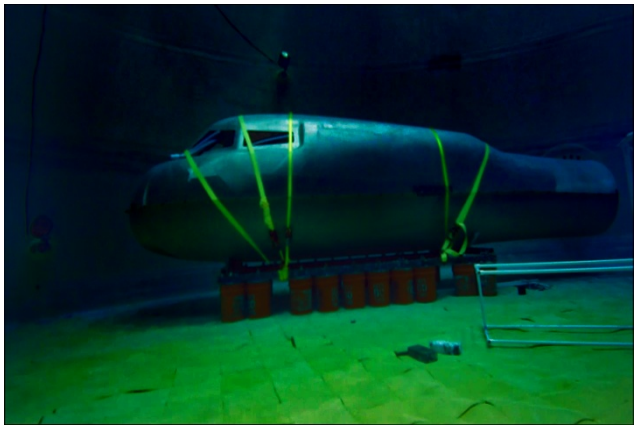


Figure 30: Underwater DCP method on Real Dataset

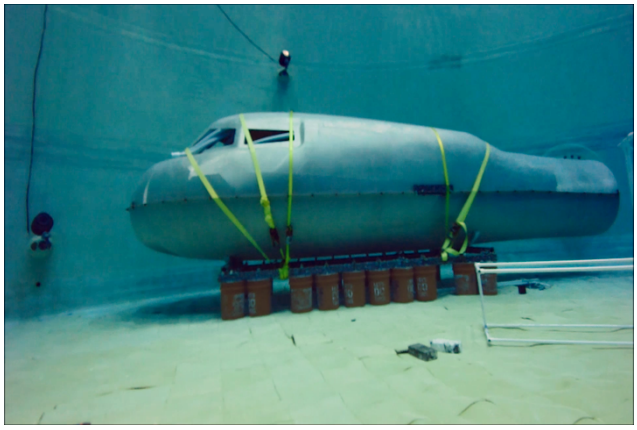


Figure 31: ULAP method on Real Dataset

Figure 32: ULAP method's TM on Real Dataset

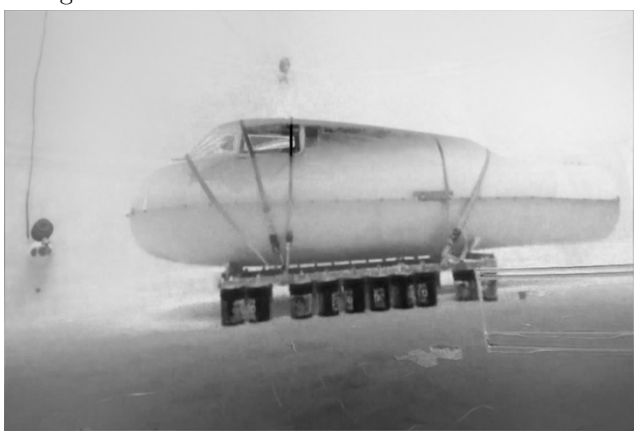


Figure 33: ULAP method's Depth Map on Real Dataset

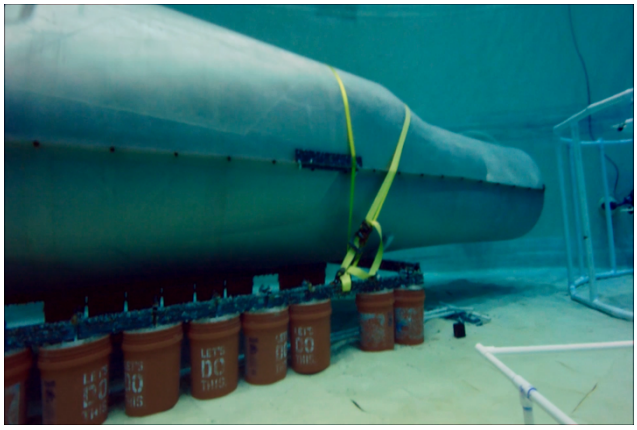


Figure 34: ULAP method on Real Dataset

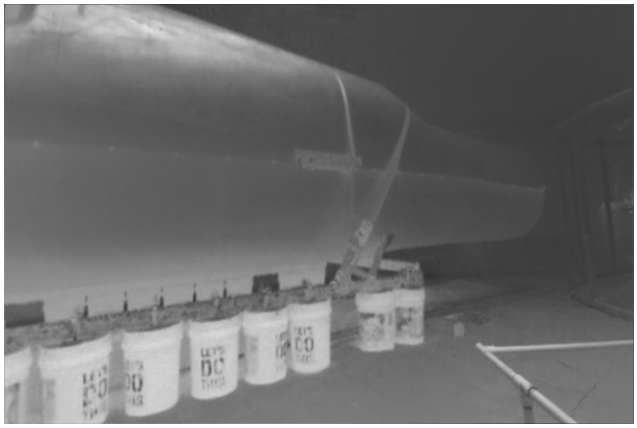


Figure 35: ULAP method's TM on Real Dataset

Figure 36: ULAP method's Depth Map on Real Dataset

Figure 37: FunIE-GAN on synthetic dataset

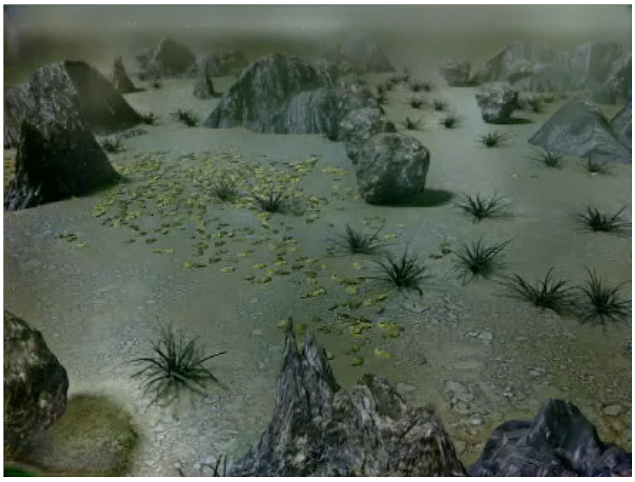


Figure 38: Wavelet-GAN on synthetic dataset

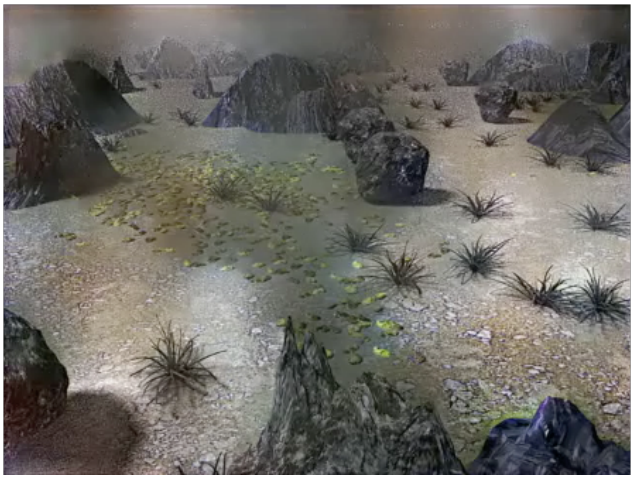


Figure 39: UIE DAL on synthetic dataset

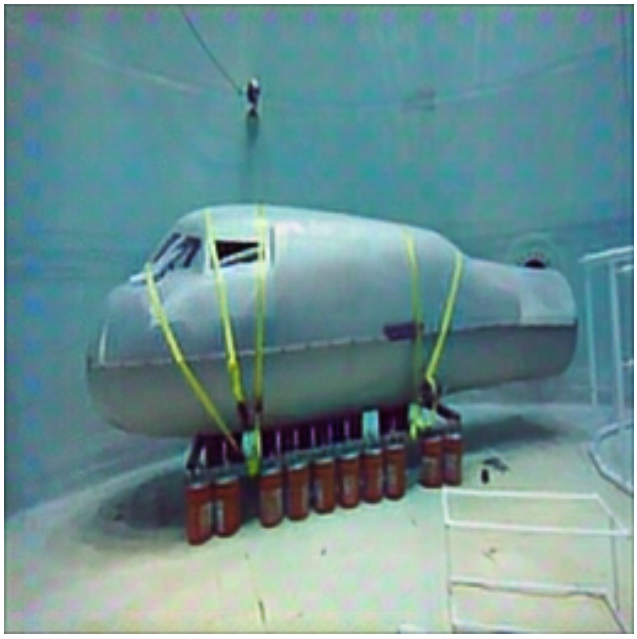


Figure 40: FunIE-GAN on real dataset

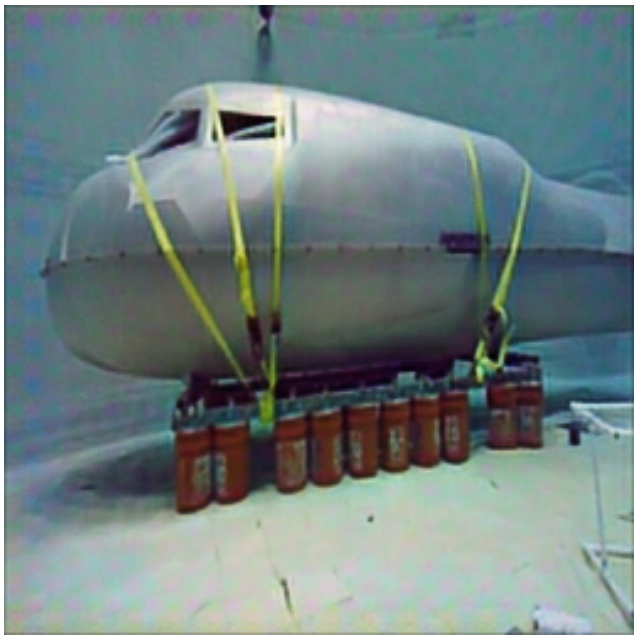


Figure 41: FunIE-GAN on real dataset

Figure 42: FunIE-GAN on real dataset

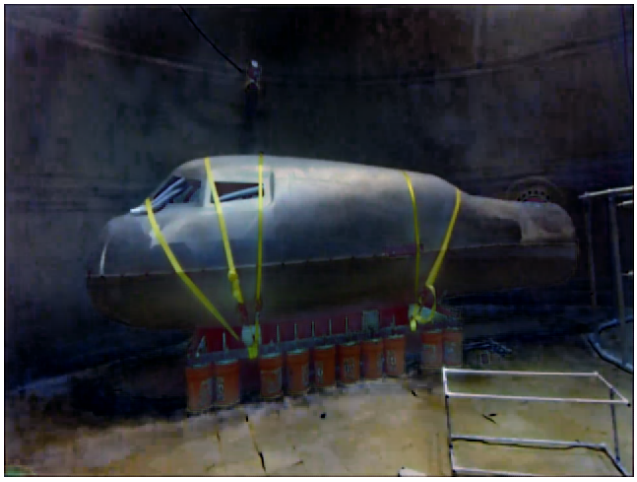


Figure 43: Wavelet-GAN on real dataset

Figure 44: Wavelet-GAN on real dataset

Figure 45: Wavelet-GAN on real dataset

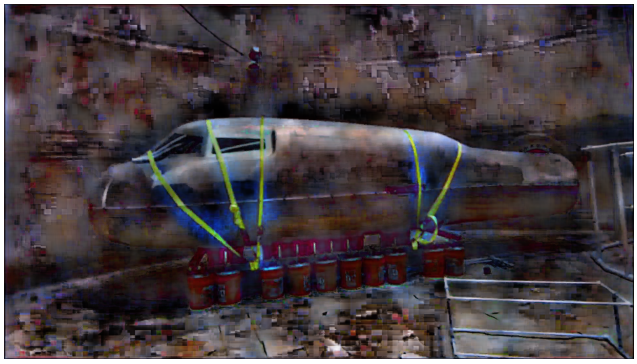


Figure 46: UIE-DAL on real dataset

Figure 47: Corrected Sample on UIEB by FunIE-GAN

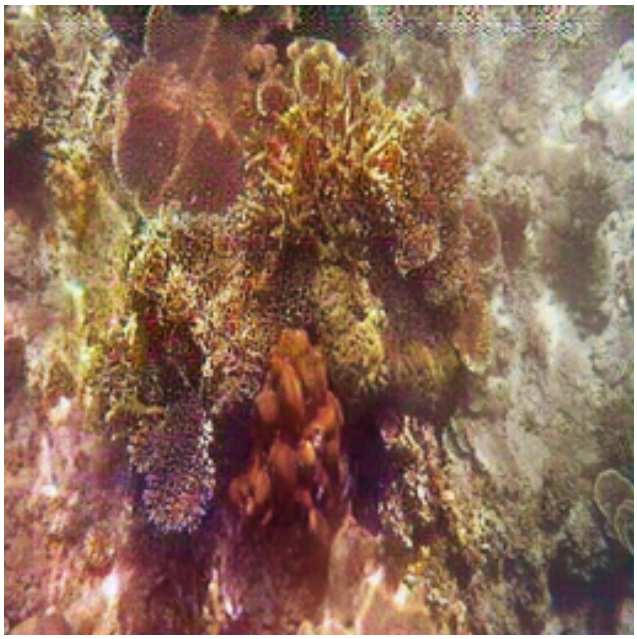


Figure 48: Corrected Sample on UIEB by FunIE-GAN

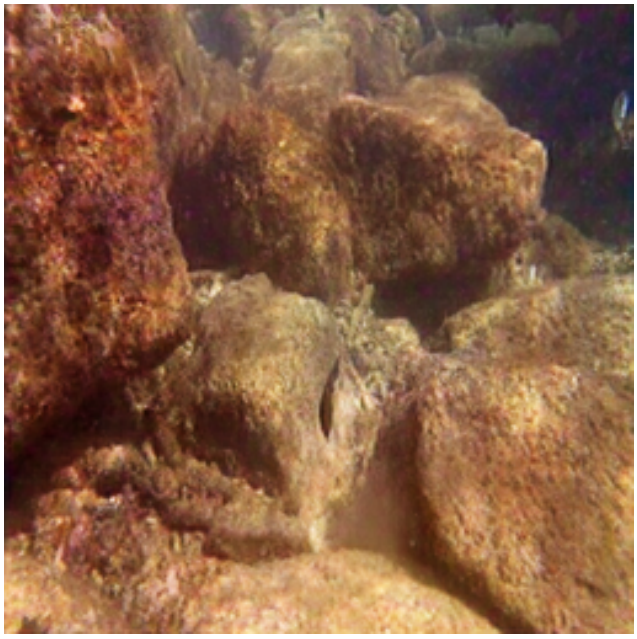


Figure 49: Corrected Sample on UIEB by FunIE-GAN

Figure 50: Corrected Sample on UIEB by Wavelet-GAN

Figure 51: Corrected Sample on UIEB by Wavelet-GAN

Figure 52: Corrected Sample on UIEB by Wavelet-GAN

Figure 53: Corrected Sample on UIEB by UIE-DAL

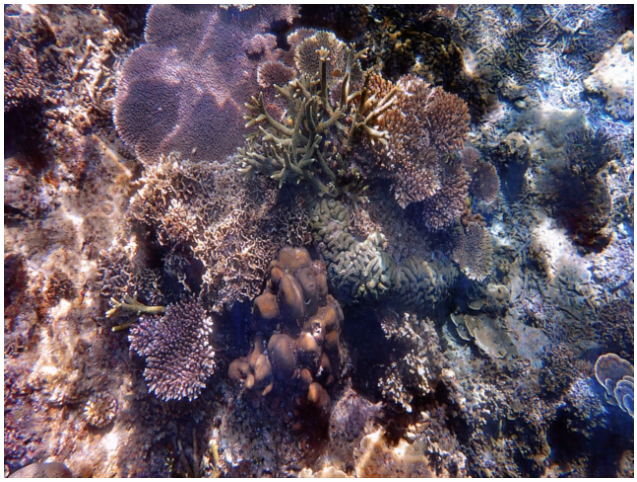
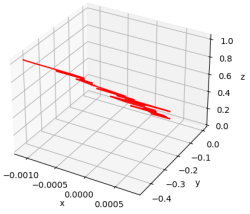
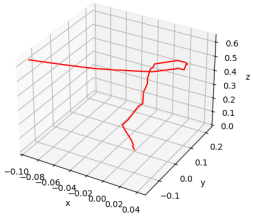


Figure 54: Corrected Sample on UIEB by UIE-DAL

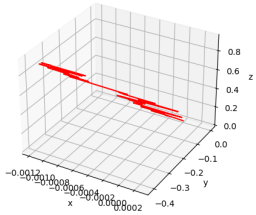
Figure 55: Corrected Sample on UIEB by UIE-DAL



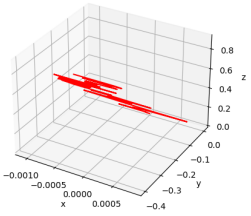
(a) Haze Line



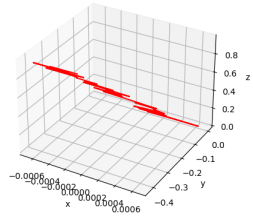
(b) FUnIE-GAN



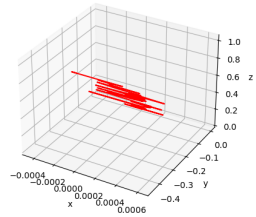
(c) Historgram Equalization



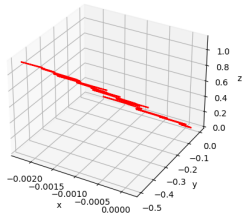
(d) HSV without Equalization



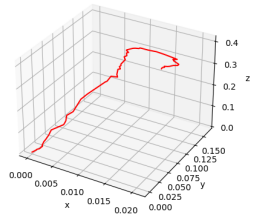
(e) HSV with Equalization



(f) original

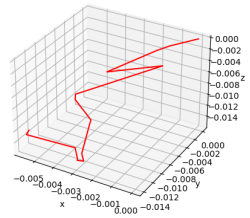


(g) Wavelet-GAN

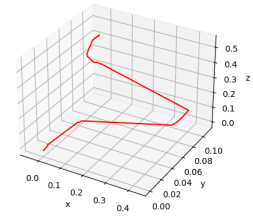


(h) White Balancing

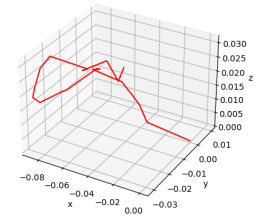
Figure 56: ORBSLAM3 results on synthetic data



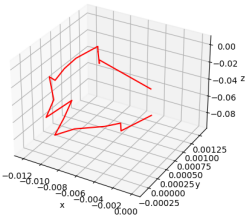
(a) Haze Line



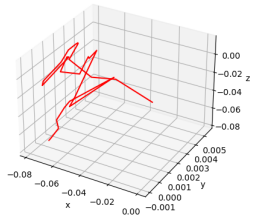
(b) FUnIE-GAN



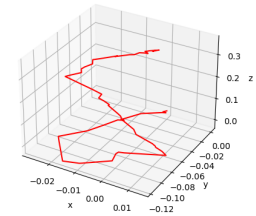
(c) Historgram Equalization



(d) IMovie enhance function

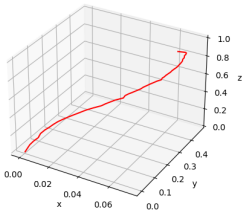


(e) Original

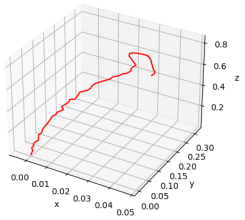


(f) Wavelet-GAN

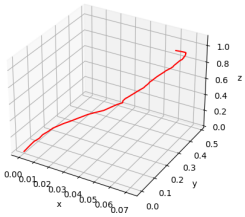
Figure 57: ORBSLAM3 results on real data



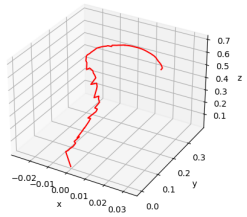
(a) Haze Line



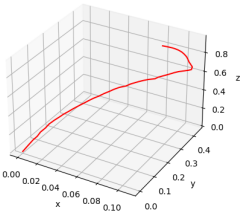
(b) FUnIE-GAN



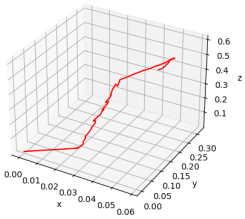
(c) Historgram Equalization



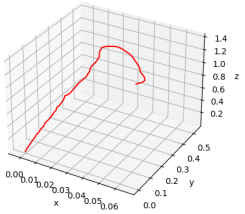
(d) HSV without Equalization



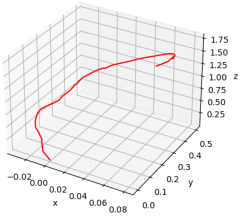
(e) HSV with Equalization



(f) Original

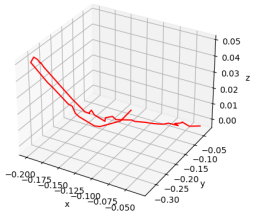


(g) Wavelet-GAN

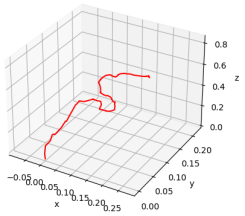


(h) White Balancing

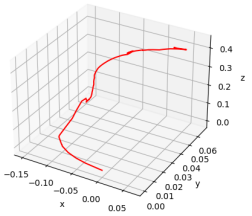
Figure 58: DSO results on synthetic data



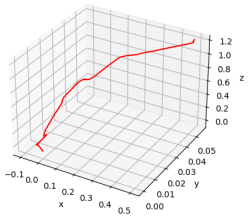
(a) Haze Line



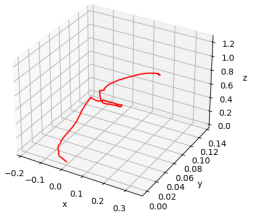
(b) FUnIE-GAN



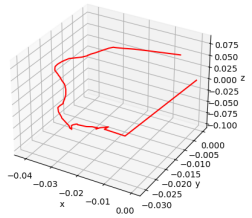
(c) Historgram Equalization



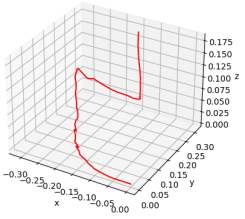
(d) IMovie enhance function



(e) Original



(f) Wavelet-GAN



(g) White Balancing

Figure 59: DSO results on real data