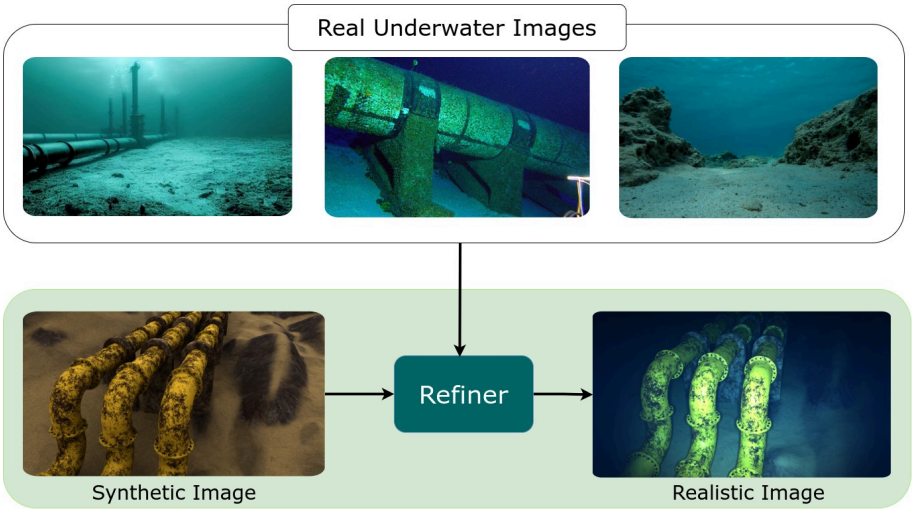
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Abdul-Kazeem Shamba

Generating Realistic Underwater Images Using Contrastive Learning

Master's thesis in Marine and Maritime Intelligent Robotics
Supervisor: Asgeir Johan Sørensen
Co-supervisor: Oscar Pizarro and Ricard Marxer
June 2023



NTNU
Norwegian University of
Science and Technology

Abdul-Kazeem Shamba

Generating Realistic Underwater Images Using Contrastive Learning

Master's thesis in Marine and Maritime Intelligent Robotics
Supervisor: Asgeir Johan Sørensen
Co-supervisor: Oscar Pizarro and Ricard Marxer
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU
Norwegian University of
Science and Technology

Generating Realistic Underwater Images Using Contrastive Learning

Abdul-Kazeem Shamba

June 14, 2023

Abstract

Artificial intelligence (AI) has gained tremendous traction in the past decade, both due to improvements in computational power and the availability of a cornucopia of useful datasets. AI techniques have seen increased application in many challenging domains such as autonomous navigation, object detection, and anomaly detection. Each of these tasks presents novel challenges and requires a high level of precision and accuracy. Most of the models we have today are supervised therefore sources of good generalization capability of the model, rely heavily on large amounts of label training data. Due to the cost, time, and complexity of providing annotations, as the natural world is unannotated, there is a need for a learning technique for generating realistic images from synthetic input while preserving the contents. Currently, the generation of realistic underwater images relies on complex physics-based models and extensive parameter adjustments, requiring domain knowledge of the underwater environment. In this thesis, we investigate the use of contrastive learning and generative adversarial network to build an end-to-end neural network for generating realistic underwater images from a set of uniform lighting synthetic inputs conditioned on real underwater images. We use the contrastive loss to preserve the content of the generated images and modify our model architecture to account for 4-channel RGBD input. After performing inference on 190 uniform lighting images from the Vision Autonomous Robots for Ocean Sustainability (VAROS) synthetic underwater data environment and computing the Fréchet inception distance (FID), we conclude that our model produced compelling realistic underwater images that can rival the physics-based images from the VAROS synthetic underwater dataset. Our statistical analysis of the FID suggests that using contrastive loss to replace the cycle-consistent loss of CycleGAN and the inclusion of depth information resulted in the highest FID when compared to other baselines for unpaired image translation. In conclusion, the creation of realistic synthetic underwater images provides numerous advantages for marine robotics practitioners and researchers. These images enable practitioners to test and optimize their specialized equipment in a controlled environment, ensuring accurate performance and functionality before deployment in real-world underwater settings.

Preface

This master's thesis on "Generating Realistic Underwater Images Using Contrastive Learning" has been written to fulfill the graduation requirement for the award of master's in Marine and Maritime Intelligent Robotics at the Norwegian University of Science and Technology, Trondheim, Norway. This thesis was developed from January to June 2023 in collaboration with the University of Toulon, Alteia Computer Vision company in France and the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. This research project has been an incredible journey, and I am grateful for the support and guidance I have received along the way. My motivation for carrying out this thesis stems from my passion for conducting research in machine learning and marine robotics. It was delightful to apply the knowledge of Software Development and ML I acquired during my education, internships, and personal projects to develop a realistic underwater image generation model. Knowing that this master's thesis is a culmination of most of my previous coursework and internships, makes this research project even more fulfilling. I have gained solid research, analytical and problem-solving skills while working on this thesis. To date, my experience as an ML engineer has been extremely rewarding and productive. However, a chance to contribute skills and theoretical knowledge in the generation of realistic underwater images on optimized hardware with good GPU resources, alongside reading recondite research papers and implementing some baseline models, afforded me the rare opportunity to conflate knowledge of Scientific Writing, ML, Robotics and software development effectively. I would like to sincerely appreciate my thesis advisors, Asgeir Sørensen, and Oscar Pizarro, for their invaluable support, expertise, and mentorship throughout the entire research process. Their guidance and constructive feedback have been instrumental in shaping the direction and quality of this thesis. I am truly grateful for their dedication and commitment. I am also grateful to my project managers, Nicola Luminari & Hacene Karrad and colleagues at Alteia who prepared me for this intricate project through my internship. I am indebted to European Union for the Erasmus Mundus Joint Masters Scholarships for providing me with the necessary resources and facilities to carry out this research. I would like to acknowledge the Department of Marine Technology here at NTNU for their support and cooperation throughout my master's program. Furthermore, I would like to express my gratitude to my diverse set of Marine Intelligent Robotics colleagues and my family who have been a constant source of motivation, encouragement, and intellectual exchange. Not forgetting the city of Trondheim for providing the "ideal" weather for thesis writing. Finally, I hope this work inspires you to explore the application of AI in the marine environment, not only can it lead to a rewarding career, but it can also create an opportunity to pioneer some unexplored domains. I hope you enjoy reading this master's thesis, as much as I enjoyed writing it!

Abdul-Kazeem Shamba
Trondheim, June, 2023

Contents

Figures

Tables

Acronyms

AI Artificial Intelligence

VAROS Vision Autonomous Robots for Ocean Sustainability

GPU Graphics Processing Unit

FID Fréchet Inception Distance

SSIM Structural Similarity Index Metric

SimCLR Simple Framework for Contrastive Learning of Visual Representations

VISSL Vision library for state-of-the-art Self-Supervised Learning

ResNet Residual Network

MLP Multilayer Perceptron

GAN Generative Adversarial Network

UIEB Underwater Image Enhancement Benchmark

MSE Mean Squared Error

RGBD Red-Green-Blue-Depth

SSL Self-Supervised Learning

VGG Visual Geometry Group

CUT Contrastive Unpaired Translation

CoDe Contrastive + Depth (Our proposed model)

SOTA State-of-the-Art

NLP Natural Language Processing

IFM Image Formation Model

ML Machine Learning

DL Deep Learning

PL Pytorch Ligntning

MLOps Machine Learning Operations

CNN Convolutional Neural Network

RNN Recurrent Neural Network

PIRL Pretext-Invariant Representation Learning

MoCo Momentum Contrast

DNN Deep Neural Network

cGAN Conditional Generative Adversarial Network

URI Uniform Resource Identifier

SDG Stochastic Gradient Descent

UUV Unmanned Underwater Vehicle

VAE Variational Autoencoder

PCA Principal Component Analysis

t-SNE t-Distributed Stochastic Neighbor Embedding

Chapter 1

Introduction

In this chapter, we provide a brief description of the problem this project aims to solve. We also provide the background as well as the motivation for this work. Furthermore, a short explanation of the main contribution of this thesis is provided followed by an overview of the thesis structure for subsequent chapters.

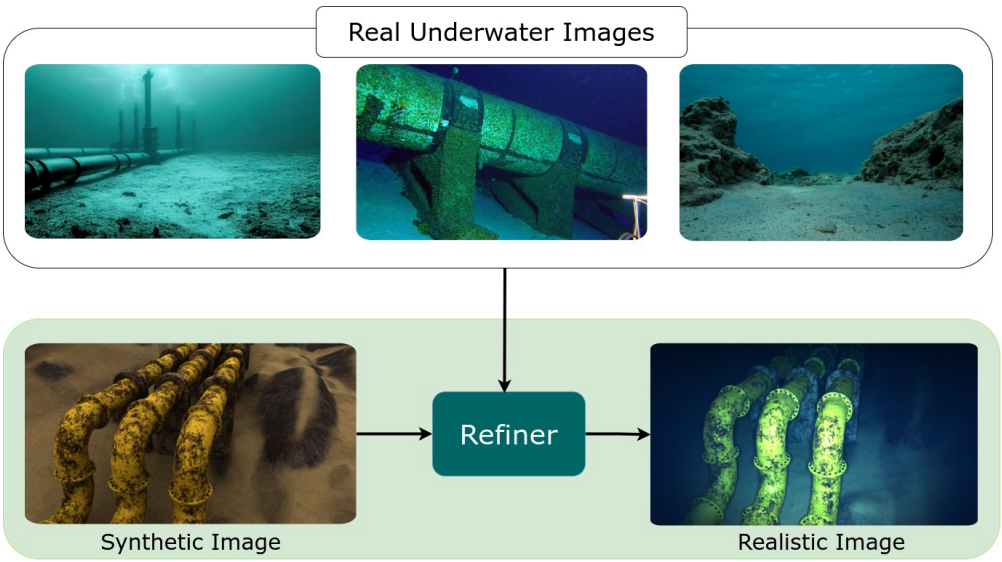


Figure 1.1: Given a uniform lighting synthetic input X and a set of real underwater images, our algorithm learns to automatically "translate" the synthetic input image to a realistic underwater image $\hat{X}$.

1.1 Background

Artificial intelligence (AI) has achieved significant improvement since its inception. Computer vision, which is an important subfield of AI, has seen some major milestones in the last few decades. With works such as optical character recognition [1], Viola and Jones face recognition [2], and the release of the ImageNet [3] dataset in 2010 which provided the foundation of many popular deep learning architectures today such as the Residual Network (ResNet) [4] and Visual Geometry Group (VGG) [5] all the way to Ian Goodfellow vanilla Generative Adversarial Networks(GAN)[6] model to more recent diffusion models [7]. Feature extraction technique in computer vision has developed from simple basis vectors as in

the Voila and Jone algorithm [2] to more robust and complex techniques of learning rich feature embeddings using deep neural networks as in the ResNet [4]. This increase in complexity has necessitated the need for more computational resources as well as the need for more labeled data, which can be difficult to obtain and sometimes require a human expert.

In the past decade, computer vision techniques using deep learning have mainly focused on supervised learning. However, since data annotation is expensive and the natural world provides vast unannotated data there has been increasing attention toward deep neural networks that learn rich semantic features from datasets without the need for annotation such as self-supervised learning (SSL). Although SSL can achieve better performance when fine-tuned on the subset of labeled datasets for downstream classification, they require a large corpus of real unlabeled data, which is often difficult to create especially in environments such as underwater. As a result of these challenges, recent works have explored the use of synthetic data to augment real-world data. These have seen real applications in underwater navigation, autonomous driving, and object detection. However, these synthetic renderings mostly have a different distribution from real images, thereby hampering the performance on downstream classification. Traditionally, the synthesis of underwater images has relied on physics-based models that require extensive parameter adjustments and domain knowledge of the underwater environment. However, recent advancements in deep learning have opened up new possibilities for generating realistic underwater images using end-to-end neural networks. Studies have underscored the efficacy of GAN networks for unpaired image-to-image translation [8–10]. We plan to exploit these GAN algorithms for generating realistic underwater images.

1.2 Objectives

The objective of this research is to develop an end-to-end neural network for generating realistic underwater images. Current methods rely on physics-based approaches that require domain knowledge and parameter tuning. We explore the effectiveness of unpaired image translation techniques to transform uniform lighting images into realistic underwater images, conditioning the network on real underwater images. The goal is to overcome the limitations of physics-based methods and provide a more efficient and accurate approach for synthesizing underwater imagery. By achieving realistic underwater image generation, this research aims to contribute to advancements in underwater image processing and enable applications in fields like marine exploration and surveillance. At the end of this master's project, the execution of the aforementioned objectives will lead us to answer the following research questions.

- **Question 1**. Can we build an end-to-end deep learning model to generate realistic underwater images with better quality or similar to the physics-based realistic underwater images without domain knowledge of the inherent underwater parameters?
- **Question 2**. Is there a way to effectively preserve the content of the uniform lighting synthetic input while retaining the style of the real underwater images?
- **Question 3**. Are there any significant improvements in using depth information as the 4th channel of our image input for underwater image generation?
- **Question 4**. What objective evaluation metric is suitable for evaluating the quality of our generated realistic underwater images?

1.3 Approach

In this research work, we aim to develop a mapping function that can translate synthetic images with uniform lighting to a new domain of realistic underwater images. Two approaches will be explored: paired image-to-image translation and unpaired image-to-image translation. The former involves training an autoencoder model and comparing its performance with state-of-the-art models for paired image translation, while the latter uses contrastive learning and generative adversarial networks to generate realistic underwater images from a uniform lighting image conditioned on real underwater dataset. The proposed methodology will be evaluated against existing techniques in unpaired image-to-image translation, and core concepts such as the objective function, model architecture, data processing, depth map, attenuation function, inference, and evaluation metric will be discussed. All experiments in this work are designed to facilitate the answering of the aforementioned research questions in Section 1.2. For example, the experiment on using a 4-channel RGBD input in our GAN model is intended to answer the **Question 4** - *Is there any significant improvement in using depth information as the 4th channel of our image input?*. The generated result from our model is evaluated via structural similarity index metric (SSIM) [11], Fréchet Inception Distance (FID) [12], and human subjective qualitative analysis and compared to other image-to-image translation baselines. We strive to ensure that every results in this master's project are reproducible. As such, we intend to publish publicly the codes, models, and datasets used for the thesis for easy validation of our results.

1.4 Contributions

Our research makes several key contributions to the field. Firstly, we propose a novel neural network architecture that is specifically designed for generating realistic underwater images. The end-to-end nature of our model eliminates the need for explicit physics-based modeling, allowing practitioners without extensive domain knowledge to generate high-quality underwater images. This not only simplifies the image synthesis process but also broadens the accessibility of underwater image generation to a wider range of users. Secondly, we investigate the efficacy of unpaired image translation techniques in the context of underwater image synthesis. By conditioning the network on real underwater images, we aim to capture the unique visual characteristics and effects of underwater environments. This conditioning enhances the realism and fidelity of the generated images, enabling accurate representation of underwater scenes and facilitating various applications in marine technology.

Overall, the contributions of this research advance the field of underwater image synthesis by providing a novel end-to-end neural network approach that effectively generates realistic underwater images. Our work enables easier access to underwater image generation for practitioners and researchers in marine technology and offers insights into the impact of conditioning on real underwater images and the integration of depth information. The findings of this research have implications for diverse applications, ranging from marine robotics and computer graphics to virtual reality, where accurate and realistic underwater imagery is crucial for immersive experiences, simulations, and scientific analysis.

1.5 Limitations

One limitation of the proposed algorithm is the failure to accurately preserve the content of the input image while transferring the underwater style in some specific cases. Although

the model successfully generates realistic underwater images, there are instances where the content of the original image is not effectively retained. This limitation suggests the need for further enhancements in content preservation, potentially through the inclusion of additional terms in the loss function to better constrain the preservation of input content in the generated underwater images. Addressing this limitation would contribute to the overall fidelity and quality of the generated results.

1.6 Outlines

The remainder of this master's thesis work has the following structure. **Chapter 2: Background.** Provides the Background and Theory in this work ranging from deep learning tools to model architecture and GPU resources. The chapter introduces the reader to all terms and concept necessary for an adequate understanding of the later chapters. **Chapter 3: Related Works.** Discusses the current State-of-the-arts (SOTA) in contrastive learning and GANs used for generating realistic data with a focus on CycleGAN [9] and CUT [10]. **Chapter 4: Proposed Architecture.** This chapter focuses on the main model architecture proposed in this work with a detailed explanation of the modification made in this thesis. **Chapter 5: Experimentation and Results.** Presents the core experiment in this work and shows the result of each experiment alongside a short analysis of the results. **Chapter 6: Discussion.** Provides comprehensive insights on the result from the previous section and underpins the significance of these results. Finally, **Chapter 7: Conclusion** Dissects the result obtained in the thesis and provides inference or conclusion where necessary. Furthermore, this chapter connects the result obtained in Chapter 5 with the research questions in Section 1.2.

Chapter 2

Background

Artificial intelligence and by extension computer vision have seen mammoth improvement in the previous years. Advances in the neural network, increase in computational power and the availability of high-quality dataset has spiraled this advancement and enabled computer vision techniques to perform exceedingly well even to the point of rivaling human experts in some specific domain application [13]. The ResNet [4] architecture has allowed models to have deeper layers without vanishing gradient. Google's "Attention is All You Need" [14] and Transformer has advanced Natural Language Processing (NLP) and vision tasks such as image captioning. In 2014, Goodfellow *et al.* [15] opened the floodgate to myriads of useful generative techniques, such as CycleGAN [9], SimGAN [8], StyleGAN [16], and a host of others. With novel applications in deep fakes, data augmentation, styles transfer, and anomaly detection. More recent techniques such as stable diffusion [17] used in Dall-e by the OpenAI team has shown promising result in image generation. Meta's latest paper on "Segment Anything" [18] underscores the development of AI techniques in the last few years. Improving the realism of synthetic images using GAN and SSL touches several techniques in AI. This chapter covers some important concepts in the field of deep learning with a focus on generative models. The tools and frameworks used for this master's thesis are also introduced briefly. These are grouped into Deep learning, Deep learning tools, SSL, GANs, Supervised learning, Dataset explanation, Evaluation metrics, and GPU resources. The authors assume the readers have some basic understanding of deep neural networks. As such, the explanations of some concepts are glossed over in the section.

2.1 Image Formation Model (IFM)

We can develop a more reliable and efficient technique to create a synthetic image that is comparable by better understanding the underwater IFM. It is well known that light behaves differently in water than it does in the atmosphere while it is propagating. When traveling through water, light is frequently absorbed and scattered. Depending on the refractive index of the material, absorption causes an energy loss when light passes through it. The propagation route is deflected as a result of scattering. Light degradation in an underwater environment is correlated with color wavelength. In reality, due to its longest wavelength, red light decays most quickly, followed by yellow and green light.

The underwater image formation model as developed by [19] and extended by [20] has three important components: direct transmission component, forward scattering component, and background scattering component, as shown in equation 2.1.

$$I_\lambda = B_\lambda + D_\lambda + F_\lambda \tag{2.1}$$

where I_λ is the total irradiance received by the camera, B_λ refers to the backscattered light when ambient light is scattered by plankton, D_λ denotes the direct light, and F_λ is the forward scattering component. Because the underwater scene and the camera are so close together, forward scattering can typically be disregarded. Therefore equation 2.1 reduces to:

$$I_\lambda = D_\lambda + B_\lambda \tag{2.2}$$

For an underwater image $I_\lambda(x)$, D_λ can be defined as

$$D_\lambda = J_\lambda(x)t_\lambda(x) \tag{2.3}$$

where $J_\lambda(x)$ is the undegraded underwater image, $t_\lambda(x) = e^{-\beta d(x)}$ refers to transmission map, β is the scattering coefficient and d(x) is the depth of scene and λ indicates the RGB color channel. B_λ can be defined as

$$B_\lambda = B(x)(1 - t_\lambda(x)) \tag{2.4}$$

where $B(x)$ is the background light. This leads to the simplified underwater IFM model

$$I_\lambda(x) = J_\lambda(x)t_\lambda(x) + B(x)(1 - t_\lambda(x)) \tag{2.5}$$

2.2 Deep Learning Tools

Training deep neural networks requires the use of certain tools or frameworks that abstracts some fundamental method. Some of these tools are libraries with already implemented SOTA architectures, whereas others are either wrapper functions for some other library or simply robust visual tools useful for hyperparameter tuning and machine learning (ML) research. This section discusses these tools and how they are used in this master thesis project.

2.2.1 Pytorch

Pytorch is an open-source machine learning framework developed by Meta, that allows developers to build and train deep learning models. It provides easy-to-use tools for creating and training neural networks, with core functionalities such as Dataloader, model architecture, loss function, and optimizers. Pytorch has gained popularity among ML researchers and developers due to its flexibility and dynamic computation capabilities. An alternative framework to Pytorch is Tensorflow developed by Google. For this thesis study, the main tool used in both the SSL and GAN network is the Pytorch framework.

2.2.2 Anaconda

Anaconda is the popular open-source distribution of Python programming language used for scientific computing, data analysis, and machine learning. It provides a rich collection of commonly used packages and tools such as NumPy, Pandas, etc We use Anaconda to create all the virtual environments used in this work. Jupiter notebook (previously called iPython notebook) which is an interactive web-based computing environment is provided by the Anaconda distribution. This notebook is used during the development and debugging

of most of the deep neural network code in this thesis work. It provides a good platform for easy data visualization and analysis. Visual Studio code is used mostly for the training process.

2.2.3 Weight and Biases

Weight and biases also known as WandB is an MLOps tool for the visualization and monitoring of ML experiments during training. It is useful for model tracking, automation, and enhancement. It helps researchers to visualize useful metrics and loss functions as well as output from the model during training. We use WandB to view the loss during SSL training and also to view the generated image at each epoch during GAN training to improve the realism of synthetic underwater data. Since WandB is a cloud-based service it is helpful in visualizing model parameters and training metrics while training on a remote computer.

2.3 Deep Learning

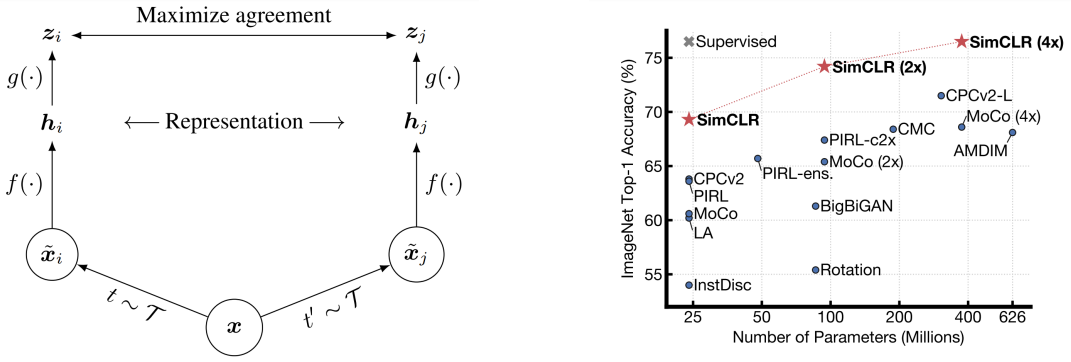
Deep learning is a subset of machine learning that involves training neural networks to learn rich feature embeddings automatically from large datasets. There is the fundamental idea that a neural network with more hidden layers can learn a more complex representation of input data and therefore achieve better performance. Deep learning's popularity in the past decade is due to its success in various applications such as object detection, natural language processing, and machine translation. Deep learning models are typically trained using back-propagation. A technique that computes the gradient in various layers used during weight updates, while minimizing the difference between prediction and actual output. Some popular neural networks are MLP, CNN, and RNN. In this section, we carefully cover some core deep learning concepts and architecture explored in this master's thesis. It is important to state here that this list is not exhaustive. For a more in-depth understanding of deep neural networks, the book by Goodfellow *et al.* [21] is a good starting point.

2.3.1 Supervised Learning

Supervised learning is the type of machine learning in which models are trained on label data, where the input data is paired with a corresponding target value. The goal of supervised learning is to learn a model to accurately predict the output given unseen test input. Supervised learning requires fully annotated training data which can be costly or difficult to obtain. The model in supervised learning is updated using optimization algorithms such as gradient descent to minimize the difference between the predicted and actual output. Supervised learning is used in this work for downstream classification tasks for Alteia's anomaly detection or fine-tuning our ResNet50 model.

2.3.2 Self-supervised Learning

Self-supervised learning is a promising subclass of unsupervised learning in which a model learns useful features from data without the explicit need for human labels or supervision. Typically the model is trained to accomplish specific tasks, known as pretext tasks. That are not directly related to the task at hand, with the hope that the model learns some useful distinguishing features that can be transferred to downstream classification tasks or fine-tuned on labeled data for supervised learning. Self-supervised learning is typically used when label data are scarce or expensive.



(a) A simple framework for contrastive learning of visual representations (SimCLR) learning algorithm

(b) ImageNet Top-1 accuracy of linear classifiers trained on representations from different self-supervised methods

Figure 2.1: A simple framework for contrastive learning of visual representations (SimCLR) and Top-1 accuracy of linear classifiers after applying self-supervised learning

2.3.3 Pretext Task

Self-supervised learning learns high-level feature embedding of data by performing some useful pretext tasks. The pretext tasks are designed in a way that is neither too difficult nor too easy for the model to accomplish and requires unlabeled data. There are several pretext task recommendations by state-of-the-art papers such as PIRL [22], MoCo [23], and SimCLR [24]. In this work, we focus on the SimCLR.

2.3.4 SimCLR

An example of a pretext task in self-supervised learning is the use of contractive learning, where a model is trained to learn embeddings that are similar when augmented samples are drawn from the same image and dissimilar for images from different samples. Simple Contrastive Learning of Representation (SimCLR) [24] is the state of the arts contrastive learning technique that has been shown to produce significant improvement in downstream classification tasks on the ImageNet dataset. This is due to its ability to learn high-quality feature representation from unlabeled data. SimCLR loss function is a variant of contrastive loss and aims to attract augmentation of similar images and repel dissimilar images. The loss function has two major components, a positive and negative term. The positive term functions to bring similar representations closer in the embedding space while the negative term does the reverse. Equation 2.9 shows the loss term with each component explained in later sections.

$$\ell_{i,j} = -\log \frac{\exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)} \tag{2.6}$$

The SimCLR uses a DNN architecture that consists of a base encoder, and a ResNet architecture, followed by a linear fully connected projection head that maps the learned high dimensional output embeddings from the ResNet encoder network to a lower dimensional space. The learning architecture of SimCLR is shown in Figure 2.1

2.3.5 Data Augmentation

Data augmentation is an important component of SSL. This technique involves taking an image sample and applying various transformations such as random crop, color jitter, grayscale, blur, horizontal flip, etc. with the aim of making the model more robust to different variations of the input data or merely increasing the quantity of the training data available. Single data augmentation can be applied as a standalone pretext task in SSL as in the case of the colonization pretext task, where an autoencoder is trained to recover an RGB image from a transform grayscale image, or as a combination of several augmentations as seen in SimCLR, where multiple augmentations are applied to the same image, to help the model in learning robust and invariant features through contrastive learning.

2.4 Model Architecture

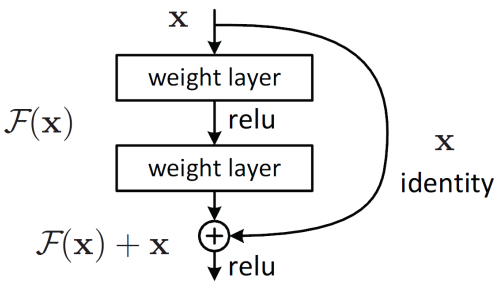
A key factor for the success in the performance of deep neural networks in a wide range of applications is the availability of multitudes of disparate models, each designed to accomplish different tasks. In this subsection, we explain some popular model architectures such as Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), ResNet, and Autoencoder. The Recurrent Neural Network (RNN) architecture has been skipped intentionally, as it does not feature in this thesis work. The GAN model architecture which is a major component in this work, is explained in the next section.

2.4.1 Multi-Layer perception (MLP)

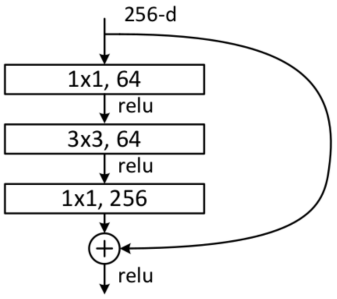
MLP is a type of feed-forward network architecture in deep learning. MLP consists of one or more layers of fully connected neurons. MLP has the ability to learn complex non-linear relationships between the input and output data, which are commonly used in supervised learning tasks, such as classification and regression. In an MLP architecture, the input is flattened and fed through a series of hidden layers with each having an activation function such as Sigmoid or ReLU to introduce some non-linearity into the model. The final layer is then passed through a Softmax as in the case of classification. For this work, given CNN's performance in extracting rich features from images, we opted for using CNN rather than MLP. However, a fully connected layer is employed in the SimCLR projection head, and while fine-tuning, the ResNet encoder for downstream classification tasks for Alteia anomaly detection.

2.4.2 Convolutional Neural Network

CNN was introduced by LeCun, Bengio *et al.* [25]. CNN uses convolutional filters to extract rich features from data and has been shown to perform exceedingly well in spatial data when compared to MLP [26]. A typical CNN architecture consists of several layers including convolutional layers and pooling layers. The convolutional layers apply a set of learnable filters to input data to extract features while the pooling layer is used for downsampling to reduce the dimensionality of the feature map and for translational invariance. The success of CNN in spatial data has seen it being applied to more complex architecture like VGG and ResNet.



(a) ResNet34 model building blocks



(b) ResNet50 model building blocks

Figure 2.2: A deeper residual function F for ImageNet and ResNet building blocks with skip connections

2.4.3 Residual Networks (ResNet)

ResNet which is short for the residual network was introduced by He *et al.* [4] to combat the problem of vanishing gradient in deep networks. Vanishing Gradient occurs during the training of a deep neural network with many hidden layers. It is a situation where the loss gradient in early layers becomes really small because they are computed using the chain rule in backpropagation. This makes updating weights in those layers difficult which inadvertently leads to slow convergence or convergence to a poor local minimum. In the ResNet architecture, this problem of varnishing gradient is subverted using skip connections between consecutive layers (See Figure 2.2a) The introduction of skipped connections function to maintain the gradient throughout the network. Providing an opportunity to scale up the number of hidden layers without vanishing gradient. The ResNet architectures are named according to the number of layers and as such we have ResNet 18, 34, 50, 101, and 152. We use ResNet50 mainly for the downstream anomaly detection work in Alteia and Figure 2.2b shows the major building blocks of ResNet50.

2.4.4 Autoencoder

Autoencoder was first introduced by Rumelhart *et al.* [27] in the paper titled "Learning representation by backpropagating errors". However, modern autoencoder architecture has evolved significantly since. Autoencoder is a special type of neural network that can learn to compress data to a low dimensional latent vector, as well as reconstruct the data. It has seen special applications in unsupervised learning tasks like dimensionality reduction, anomaly detection, data denoising, and self-supervised learning pretexts. In general, Autoencoder architecture consists of two major parts as shown in Figure 4.2. An encoder that maps the input to a low dimensional vector space known as a latent vector, and a decoder that reconstructs the original input or some variation of the input data. The encoder and decoder networks are trained simultaneously to minimize the reconstruction error between the original input and reconstructed output.

2.4.5 Generative Adversarial Network (GAN)

The GAN forms the backbone of this master's work and is covered comprehensively in the next chapter. Depending on the application, there are several sub-variant of the generative adversarial networks. For example, the vanilla GAN can generate random images occupying

Table 2.1: Folder naming and corresponding content of the VAROS dataset

Folder	Contents
A	Underwater RGB Image
B	Uniform Lighting Image
C	World Space Normal Image
D	Depth Map Image

similar distribution as the training data given a random noise as the input. However, this model has a major limitation as the user has little or no control over the output category. This shortcoming led to the development of Conditional GAN (cGAN) by Mirza and Osindero [28] which takes a noisy input and a class to condition the generated output on a specific class. Other variants that have gained popularity are the StyleGAN [16] and CycleGAN [9] discussed in the next section. The replacement of CycleGAN cycle-consistent loss with a contrastive loss is an improvement that we explored in this work.

2.5 Dataset

A major contributor to the success of the ML model recorded in recent years has been the availability of large training datasets with hundreds of thousands of training examples. Creating these examples with adequate annotations for supervised learning is often expensive, especially in domains like underwater where conditions are harsh. The difficulty in obtaining labeled datasets is the fundamental purpose of this research work in the first place. The good news is that both models developed in this work are categorized as unsupervised learning and as a result do not require any need for explicit labels or annotations. The dataset used in this master's thesis work covers different categories, ranging from the custom dataset from the Alteia Datalib platform to the synthetic images from the VAROS [29] environment. Finally, the Underwater Image Enhancement Benchmark (UIEB) [30] dataset was used for the unpaired image-to-image translation. This section briefly explains these two categories of the dataset including the distinguishing elements in all datasets and their specific use case in this project.

2.5.1 Vision Autonomous Robots for Ocean Sustainability (VAROS)

The VAROS data developed by Zwilgmeyer *et al.* [29] is a synthetic underwater dataset developed as part of the computer vision wing of the Autonomous Robots for Ocean Sustainability. The environment used is a 3D reconstruction of the Vasquez Rocks. Table 2.1 also shows the folder naming structure of the VAROS environment and the content of the folder. For this master's thesis, our focus is on the following folders A, B and D.

For a better understanding of the overall folder structure see Figure 2.5.1 The VAROS dataset provides a rich source of underwater sequence data that is useful for simulating underwater navigation and enhancing underwater images. However, the majority of this dataset are samples from the same environment and therefore lack a variety of features of the underwater environment. An example of this environment is shown in Figure 2.3. Thus, a model trained on this dataset may fail to generalize to other underwater scenery. The VAROS dataset contains 4715 RGB underwater images and their corresponding non-underwater uniform lighting images.

```
cam0
```

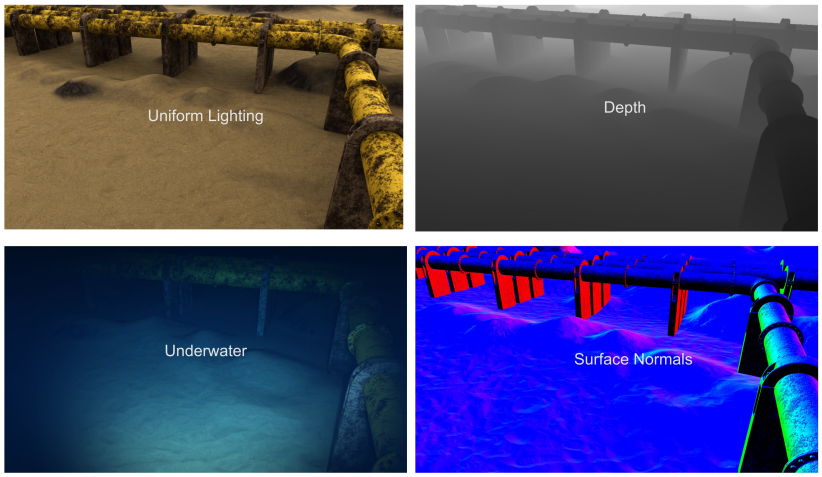


Figure 2.3: Image content of the VAROS synthetic underwater dataset with the uniform input images used during inference for our model.

```
├── A
│   ├── seq01_veh0_camM0_A-00000000.png
│   ├── seq01_veh0_camM0_A-00000001.png
│   ├── ...
│   └── seq01_veh0_camM0_A-00004714.png
├── B
│   ├── seq01_veh0_camM0_B-00000000.png
│   ├── seq01_veh0_camM0_B-00000001.png
│   ├── ...
│   └── seq01_veh0_camM0_B-00004714.png
├── C
│   ├── seq01_veh0_camM0_C-00000000.png
│   ├── seq01_veh0_camM0_C-00000001.png
│   ├── ...
│   └── seq01_veh0_camM0_C-00004714.png
├── D
│   ├── seq01_veh0_camM0_D-00000000.png
│   ├── seq01_veh0_camM0_D-00000001.png
│   ├── ...
│   └── seq01_veh0_camM0_D-00004714.png
├── camM0_poses
│   ├── camM0_poses_euler.csv
│   ├── camM0_poses_quaternion.csv
│   └── camM0_poses_transformation_matrix.csv
├── camM0.yaml
└── camM0_timestamps.yaml
```

2.5.2 Underwater Image Enhancement Benchmark (UIEB)

The Underwater Image Enhancement Benchmark (UIEB) is a standardized evaluation framework specifically designed for assessing the performance of underwater image enhancement

algorithms. It serves as a benchmarking tool to compare and analyze different methods used for enhancing the quality and visual appearance of underwater images. UIEB provides a curated dataset of underwater images, which typically suffer from various challenges such as color distortion, low visibility, and lack of contrast. These challenges arise due to the inherent properties of water, including light absorption, scattering, and color attenuation. We use the UIEB dataset and our custom real underwater dataset to condition our model during the learning process.

2.6 Loss Function

In deep learning, the loss function plays a crucial role during the training process. The selection of the loss function is highly dependent on the model architecture and data being processed. Models are trained to minimize the loss function by computing the difference between predicted and target values. During backpropagation, the gradient of this loss function with respect to the weights is computed and used for the weight update optimization step using a learning rate and a weight update process such as the stochastic gradient descent (SGD). In large neutral networks with high dimensional datasets, the loss function is usually a complex manifold with local minima, saddle points, and global minimum. Since convergence to a global minimum is mostly intractable, the optimization objective in large networks is to find a suitable local minimum. There are myriads of loss functions available today depending on the task. In this subsection, we cover three (3) popular loss functions that have been explored in this research work.

2.6.1 Cross Entropy

Cross entropy loss is commonly used in deep learning for classification tasks. Typically, this loss is applied after the Softmax activation function that maps the output of the final fully connected layer to a probability distribution over classes, ensuring that the predicted probabilities sum up to 1. Cross entropy loss measures how well the model's predictions match the ground truth labels. This loss value is non-negative and a lower value signifies better model performance. Equation 2.7 shows how the cross entropy loss is computed.

$$\text{Loss}(y_i, \hat{y}_i) = -\sum_{i=1}^{\text{C}} y_i \cdot \log \hat{y}_i \tag{2.7}$$

Where L is the cross-entropy loss, y_i is the ground truth probability distribution, C is the total classes, and $\hat{y}_i$ is the predicted probability distribution. We use this loss in this project for downstream classification tasks aimed to evaluate the performance of our SSL approach.

2.6.2 Mean Square Error

Another very common loss function used in deep learning is the MSE loss. This loss has seen significant application in regression tasks. As the name implies, it measures the average square difference between the predicted and actual value of the output variable and is given by Equation 2.8

$$\text{L}(\text{y}_\text{i}, \hat{\text{y}}_\text{i}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.8}$$

Where L is the MSE loss, N is the number of samples, y_i is the actual output and $\hat{y}_i$ is the model prediction. In this thesis project, this loss function is applied between the expected output image and the model's prediction image. In this case, N is the dimension of the output image and the individual pixel is seen as a continuous value, hence the need for a regression loss to measure how well the values match the expected ground truth.

2.6.3 Contrasive Loss

The contrastive Loss function is designed to maximize the agreement between representations from similar samples while minimizing the similarity between representations of dissimilar instances. The SOTA paper SimCLR uses a variant of contrastive loss function called the NT-Xent (Normalized Temperature Scaled Cross-Entropy) loss. This loss also has some derivation from cross entropy. According to the SimCLR paper, a contrastive prediction task is derived by applying augmentation to N examples resulting in a 2N minibatch. The authors do not sample negative examples explicitly but rather given a positive pair, the other 2(N-1) augmented examples are treated as negative examples. The loss function for a positive pair example according to the paper is shown in Equation 2.9.

$$\ell_{i,j} = -\log \frac{\exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)} \tag{2.9}$$

Where $\mathbb{1}_{[k\neq i]} \in \{0, 1\}$, z_i and z_j represent the positive pair vector after the projection layer and τ denotes the temperature parameter

2.7 Evaluation Metrics

Evaluation metrics are useful in measuring the performance of a given model. Compared to the loss function, evaluation metrics provide a clear indication of a model's performance. Furthermore, the evaluation metric provides a suitable benchmark for comparing models. Although loss functions are useful for optimizing models parameter and providing insight into the learning process, they do not provide a benchmark for comparing model performance. The value of a loss can be dependent on several hyperparameters such as the hidden layers, batch size etc, and are rarely a direct indication of a model's performance. It is worth noting that evaluating the output of a GAN can be very challenging as there is no straightforward objective metric that can be used to access the quality of the generated output. As a result, the most commonly used method for evaluating the output of a GAN network is visual inspection. This involves a human evaluator providing valuable insights into the overall quality, realism, or diversity of the generated samples. In addition, we use objective quantitative metrics such as the FID and SSIM to augment our subjective evaluation (See section 5.2.2).

2.8 GPU Resource

The availability of powerful GPU resources has had a tremendous impact on the performance of ML models in recent years. GPUs provide significant advantages over CPUs, both in computational speed and efficiency. GPU offers a robust unit, capable of handling parallel computation, which makes it suitable for most neural networks with millions of simultaneous computations. Also, GPUs are highly energy efficient and consume less power making

Table 2.2: Names and Specifications of the remote machines provided by the Alteia data science team.

GPU Specifications					
GPU Name	CPU	Speed	GPU	RAM	Disk
Data2	Intel(R) Core i7	3.50GHz	1080	32GB	2TB
Data3	Intel(R) Core i7	3.80GHz	1080	32GB	3TB
Calypso	AMD Ryzen 3960X	24-Core	3090	128GB	2TB

them a better cost-effective option in deep learning. The following section describes the GPU resources used for this thesis work.

2.8.1 Calypso and Data2

The main GPU resources used in training the deep learning model in this study are the remote computer provided by Alteia. The GPU computers named Calypso and Data2, were assessed remotely by a "ssh" command to their individual IP addresses with a VPN. The TigerVNC viewer is used to provide a GUI to either of these remote machines. The individual specifications of the GPU resources offered by Alteia are shown in Table 2.2

2.8.2 Google Colab

The Google Collaboratory is a cloud-based platform that is provided on the Jupiter notebook environment to help researchers, developers, and students to build ML models using GPU resources. Google Colab offers the Tesla K80 and Tesla T4 as free GPU resources to users. With a monthly subscription fee of 10 euros, users can have access to the more powerful GPU resources such as the Tesla V100 Tesla P100. In this study, we use Google Colab occasionally to train our model when having dependency conflicts with packages in the remote computer.

Chapter 3

Related Works

This chapter introduces previous works on the generation of realistic underwater images. They are broadly categorized as physics-based or mathematical models and deep neural network approaches.

3.1 Physics-based Image Synthesis

Underwater image formation is a complex process influenced by factors such as reflectance, backscatter, light propagation, and refraction. Understanding the underwater image formation model is critical to creating realistic underwater rendering. The Image Formation Model (IFM) as discussed in Section 2.1, is often represented in the form of a mathematical model. Through the exploration and application of underwater IFM, researchers have made progress in creating realistic-looking underwater renderings [31]

The authors in [32] explored using modern computer graphics tools such as ray tracing and physically accurate models of seawater to run a simulation of the underwater environment of how light is transmitted and captured by the image sensor. The author shows that adjusting the model parameter can help in predicting depth information from the reference target of known spectral reflectance. Álvarez-Tuñón *et al.* [33] proposed a framework that effectively combines post-processing by a UUV simulator [34] with the linearized formula suggested in [20] and style transfer technique to create realistic underwater images. According to the authors, their approach creates a realistic underwater environment suitable for underwater navigation simulation. In [31], the authors created an underwater simulator by extending the Jaffe-McGlamery [20] model to include multiple light sources, render shadows, and a parameterized volume scattering function. The proposed method is focused on deep-sea scenarios devoid of natural light. The authors tested the realism of the proposed simulator bu building a synthetic model of an underwater scene and adding a cube with a checkered board pattern.

A more recent work by Song *et al.* [35] titled "Deep Sea Robotic Simulator" presents a physical model-based solution that combines in-air texture and depth information to generate deep sea underwater image sequences. The authors acknowledge the effect of artificial illumination in the deep sea image formation model to account for both attenuation and backscattering effects. The VAROS synthetic underwater environment developed by Zwilgmeyer *et al.* [29] aims to create a highly realistic scene rendering using the ray tracing method while integrating direct light and indirect volumetric scattering. The use of the raytracing method enhances the capability of this technique in producing images with accurate illumination. One thing that all the aforementioned methods have in common is

that they require domain knowledge of the inherent underwater properties and parameters present in the IFM. Another obvious limitation is that some inherent optical properties of water such as absorption and scattering coefficient are dependent on the wavelength, therefore require fine-tuning and parameter adjustment during application.

3.2 Autoencoder Image Generation

The limitations of the previous physics-based method have spurred researchers to develop an end-to-end network that automatically learns to generate underwater images without the need for knowledge of the inherent optical properties of the underwater environment. Autoencoder as discussed in Section 2.4.4 is one such technique used for end-to-end image reconstruction.

In [36], the authors used the Unet denoising end-to-end autoencoder for underwater color restoration. Since this approach requires paired data, they developed new data that combines different underwater parameters such as turbidity depth, temperature, and attenuation for training their proposed network. The method used to generate the synthetic data is similar to the work by Fabbri *et al.* [37]. According to the authors, the proposed method has good generalization and high performance in restoring the color of degraded underwater images while preserving image details. Yu and Qin [38] also proposed an end-to-end underwater image enhancement framework that combines fractional integral-based Retinex using an unsupervised autoencoder network that integrates an advanced attention mechanism and a modified loss function. The Retinex is useful in modeling objects in the image with a particular reflection performance. The paper applied their approach in a comprehensive analysis of 3 underwater datasets.

Kim *et al.* [39] in their 2021 paper took a shift from the underwater environment by proposing a novel dehazing approach using the Wasserstein autoencoder. Compared to a conventional autoencoder (AE) with low dimensional latent vectors, their approach learns a 2-dimensional latent tensor and matches the haze inputs with the dehazed output in a pixel-wise way. In addition, the authors presented a tree-structured fusion technique to enrich the feature representation of latent tensors. A special feature of this paper is the application of WAE on different conditions such as low-light, day and night and underwater to buttress the robustness of the proposed algorithm. Autoencoder sometimes produces ambiguous reconstructions that occupy different distributions from the input, hence the reason for the development of Variational Autoencoder (VAE) [40] to impose structure on the latent space. As a consequence, Xu *et al.* [41] proposed a novel unregularized adversarially approximated autoencoder (AAAE) to approximate the implicit probability distribution. The methods described above are all used for dehazing or enhancing underwater images which is different from our task in this work. Our task is creating realistic underwater images which can be viewed as an inverse problem to dehazing or underwater image color correction. There is a paucity of research available on using autoencoder for the generation of realistic underwater images. This lack of research can be attributed to several factors. The first is that image restoration and enhancement is a more popular problem. Also, the use of a denoising autoencoder to generate realistic underwater images requires paired training examples of in-air and underwater images which are difficult to obtain in real life. However, with the rapid development of simulation tools, creating realistic underwater data is beginning to gain popularity [42, 43]. Due to this gap in research, one of our proposed architectures in the next section is to build an end-to-end autoencoder training pipeline trained on paired samples of underwater and in-air images from the VAROS dataset, to investigate the gener-

ative capability on unseen data and compare the results with the Pix2pix method that also uses paired training examples.

3.3 Neural Style Transfer

Style transfer has its origin in non-photorealistic renderings and is closely related to texture synthesis and transfer. Neural style transfer is a technique that effectively combines the content of one image with the style of another to create a new image. This technique was introduced by Gatys *et al.* [44] in the popular paper "Image Style Transfer Using Convolutional Neural Networks". By minimizing the content and style loss simultaneously, the neural style transfer algorithm synthesizes a new image while maintaining the artistic style of the style image.

The fundamental idea behind style transfer is that an image style and content are separable entities [44]. Thus it is possible to change the style of an image while preserving its content. Contrary to a typical neural network training that applies a gradient on the weights of a model, in style transfer the gradient is applied on a white noise image that is synthesized to capture the style and content of two input images. In an underwater setting, the content of the image can be the features of the image as it would appear in-air such as the terrain, corals, etc while the style would factor in the color, backscatter, reflectance, or any optical parameters that are usually manipulated using the IFM. However, this approach of applying a gradient on a single image during inference is limited and inefficient, especially in real-time applications such as applying new styles to video. What we want to learn is a model that can instead apply an underwater realistic style to an image or sequence of frames in real-time efficiently.

Neural style transfer has seen good applications in various domains such as artistic renderings in various domains including underwater. Lee *et al.* [45] applied style transfer on underwater sonar images given a depth image obtained from a simulator. In the paper, a water tank and sea style were considered and a network with a style bank is learned to perform style transfer while optimizing the losses in [46]. In [47], the authors designed a method that uses the style transfer technique by converting sonar images into optical styles to enhance the matching performance of underwater sonar images. Some papers such as [16] build upon the idea of neural style transfer to develop a GAN-based model. We plan to apply the neural style transfer on a few paired samples of images in this work to show how this method compares to others.

3.4 Image to Image Translation

Image-to-image translation is a machine learning problem that given a set of aligned aims to learn a mapping function between the input and output image.

The generative adversarial network since first proposed by Goodfellow *et al.* [15] has seen some remarkable applications in image-to-image translation. The vanilla GAN takes in as input a random noise and as such is deficient in changing the domain of an input. The paper by Gatys *et al.* [44] has inspired several works in GAN such as styleGAN [16], StarGAN [48], and CycleGAN [9]. In the underwater environment, GAN is commonly used to create a realistic underwater dataset for training an end-to-end image enhancement model [49, 50]. StyleGAN motivated by the work of Gatys *et al.* [44] designed a generator architecture and controls the image synthesis process by adjusting the style of a learned constant input at each convolutional layer. Their architecture also explores the unsupervised separation of

high-level features. Pix2pix [51] is a powerful paired model that proposed the use of cGAN to learn a mapping function from the input image to a target image.

To achieve unpaired image-to-image translation, the popular CycleGAN [9] architecture learns a mapping function such that the distribution of the output from the mapping function when applied to the input image is indistinguishable from that target image. CycleGAN uses two generators and two discriminators to translate images in two domains and proposes the cycle-consistent loss to tackle the issue of mode collapse in unpaired image translation.

In [8], the authors in a bid to reduce the gap between synthetic and real image distribution proposed a SimGAN to learn a model to improve the realism of a simulator's output conditioned on unlabeled real images, while preserving the annotation information of the simulator. This work aligns closely with what we aim to achieve in the work. We propose a similar technique in the underwater environment by learning a refiner model to generate realistic underwater renderings given input with a synthetic image with uniform lighting and depth map conditioned on a set of real images as shown in Figure 1.1.

Li *et al.* [50] proposed the WaterGAN architecture. This paper uses a two-step approach. The first step is to generate realistic underwater images by learning a GAN model using in-air and depth map pairings and the second step involves using the generated underwater images to train a restoration autoencoder. A major limitation of this technique is that it can be viewed as a color transfer technique rather than improving the realism of synthetic underwater renderings. This is expected as the WaterGAN is trained using real images rather than synthetic ones and as such finds no explicit need to improve the realism. Our approach improves on this by taking as input a synthetic image and exploring ways to improve the realism and replicate the color of underwater renderings. Precisely, our approach can be seen as a conflation of the SimGAN and WaterGAN methods.

Chapter 4

Proposed Architecture

The core objective of this research work is to generate realistic underwater images from synthetic uniform lighting renderings. In a broader sense, this task is seen as image-to-image translation i.e. translating an image from one domain to another. This translation can be applied directly to the image pair in the case of Gatys *et al.* [44] style transfer or more broadly learning a model that performs this mapping on any set of images. In this work, the two domains are uniform lighting synthetic images and realistic underwater images. For example, given a collection of synthetic images x we plan to learn a mapping function that can translate to a new domain of realistic underwater images. Image-to-image translation has been chiefly solved using two techniques (I) paired image-to-image translation and (II) unpaired image-to-image translation. Due to the difficulty in obtaining paired training examples, much of recent work has focused on unpaired image-to-image translation [8–10, 52]. In this thesis work, we explored both techniques. First, we build a rather naive image translation model using autoencoder (Figure 4.2) and compare our results with those obtained using the state-of-the-art pix2pix [51]. Second, we adapted the CUT [10] framework and adjust the input to account for an extra dimension of depth information. We call our method CoDe (Contrastive + Depth) for simplicity. Figure 4.1 shows an overview of our CoDe algorithm. We perform a comprehensive comparison with the state-of-the-art in unpaired image-to-image translation as shown in Chapter 5. The remainder of the Chapter explains some core concepts such as the objective function, model architecture, data processing, depth map, attenuation function, inference, and evaluation metric.

4.1 Objective Function

Objective function is a fundamental component in deep learning it provides information on the model learning and is used in the optimization process. The overarching goal in a deep learning training process is to learn a model to minimize the objective or loss function in search of a global or local minimum. The choice of an appropriate objective function is crucial in the learning process and directly affects the optimization process and determines the direction and magnitude of the gradient steps. For example, in the classification task, cross-entropy loss is commonly used while the MSE or mean Square error is used in the regression task. In this work, the autoencoder image generation is seen as a regression task, and as such we use the MSE loss.

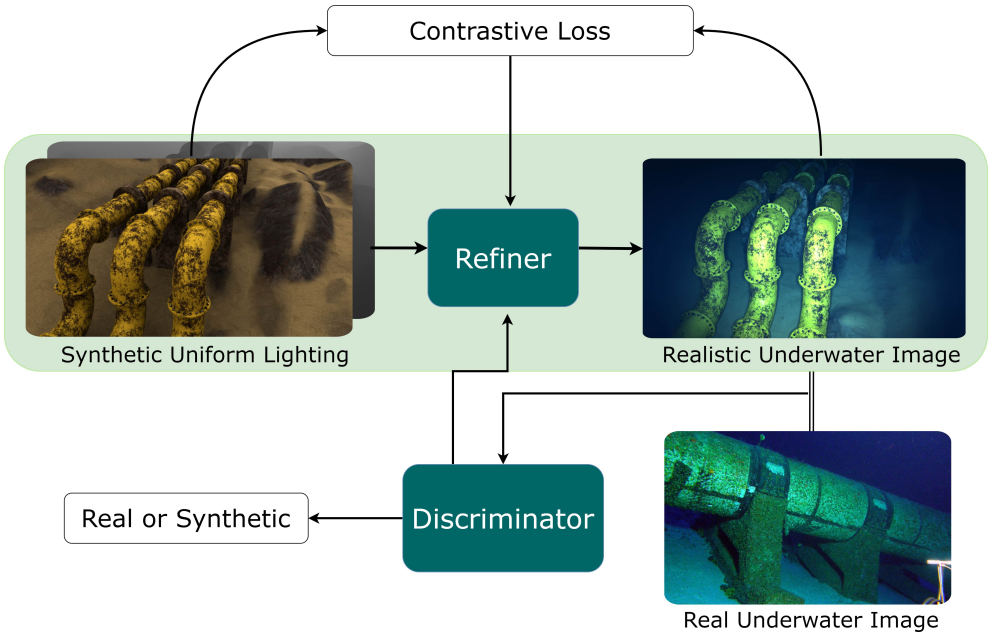


Figure 4.1: A realistic underwater image generation learning procedure using contrastive learning and depth maps (CoDe). The Refiner is our model generator and conditioned on real underwater images while using the contrastive loss to preserve the content of the synthetic uniform lighting input

4.1.1 Mean Square Error

We apply the MSE loss in our autoencoder architecture to minimize the pixel-wise mean square error between the generated underwater images and the expected ground truth in a paired image-to-image translation technique. The MSE can be expressed as

$$\mathrm{L}(\mathrm{y_i}, \hat{\mathrm{y}}_\mathrm{i}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.1}$$

Where L is the MSE loss, N is the dimension of our image data andy_i and $\hat{y}_i$ are the expected and generated underwater images respectively. A major limitation of the MSE loss is that it focuses on pixel-level similarities and does not consider the perceptual quality of the image. An alternative to the MSE loss is the SSIM. SSIM captures the perceptual quality between two images by considering intensity, contrast, and structural information

4.1.2 Adversarial Loss

The adversarial loss is useful in making the output of our generator to be as realistic as possible in comparison to real underwater images. Giving an unpaired instance of $\mathrm{X} = \{\mathrm{x} \in \mathscr{X}\}$ and $\mathrm{Y} = \{\mathrm{y} \in \mathscr{Y}\}$, we plan to map images from input X to output Y while preserving the content of domain $\mathscr{X}$. The objective function of GAN is shown below

$$\begin{aligned}\mathcal{L}_{\mathrm{GAN}}(G, D, X, Y) = \mathbb{E}_{y\sim p_{\mathrm{data}}(y)}[\log D(y)] \\ + \mathbb{E}_{x\sim p_{\mathrm{data}}(x)}[\log(1 - D(G(x))]\end{aligned} \tag{4.2}$$

The aim of the generator G in our case is to generate realistic underwater images that make it difficult for the discriminator D to distinguish. This is seen as a zero-sum game in the

minimax fashion. Where G tries to minimize the objective function against an adversary D that aims to maximize the objective function. Previous techniques such as SimGAN [8] have combined the GAN loss with more traditional loss such as the $L1$ loss. With the additional loss, the generator functions not only to generate realistic output but also to be as close as possible to the synthetic input. SimGAN enforces this in their approach with a regularization parameter Λ to preserve the annotation information of the synthetic input images. In [9], two generators and two discriminators were introduced. $G : X \rightarrow Y$ for mapping X to Y and $F : Y \rightarrow X$ for mapping Y to X. The authors applied these two generators in the cycle consistency loss to tackle the famous GAN problem of mode collapse and to reduce the space of possible mapping functions. This cyclic translation has been shown to produce remarkable results [9]. However, in this work, we adapted the approach from [10] which uses patchwise contrastive loss. This method only requires a single generator and discriminator as it involves mapping in one direction, which captures the need of this thesis work. A single generator and discriminator can remove complexities from the training procedure as well as reduce training time.

4.1.3 Contrastive Loss

We adapted the method in [10], as mentioned in the foregoing, this approach is similar to the CycleGAN but uses patchwise contrastive loss to maximize the mutual information between the input and output for one-sided image translation. Contrastive loss has been explained in detail in Chapter 2, so we will not go into much detail here. In the context of image translation, we use the contrastive loss on the synthetic uniform lighting input and generated realistic underwater image Y. We first reuse the encoder part of our generator architecture to map input X and output Y into the feature space. Then we sample a query patch from the generator output y and compare this to a patch from the same location in a contrastive learning approach. Just like in the case of SimCLR, a two-layer MLP network is used as the projection head. The distance between the query and other examples is scaled by temperature $\tau = 0.07$. The idea of using contrastive loss to improve the realism of synthetic images is simple. For this situation, it is expected that the input and the output images should have the same content. For example, given a patch that captures a coral reef in the output image we expect to be able to associate this with a corresponding coral reef part in the input more than other parts of the underwater terrain or seafloor. The authors in [10] name this loss PatchNCE as shown in Equation 4.3

$$\mathcal{L}_{\text{PatchNCE}}\left(G,H,X\right) = \mathbb{E}_{\boldsymbol{x}\sim X} \sum_{l=1}^{L} \sum_{s=1}^{S_l} \ell\left(\hat{\boldsymbol{z}}_l^s, \boldsymbol{z}_l^s, \boldsymbol{z}_l^{S\setminus s}\right). \tag{4.3}$$

Where $\boldsymbol{x}$ is the layer of Interest, $\boldsymbol{S}$ is the number of spatial locations in each image patch. $\hat{\boldsymbol{z}}_l^s$ is the embedding space of the output $\boldsymbol{y}$ the corresponding positive query is $\boldsymbol{z}_l^s$ and other negative features are represented as $\boldsymbol{z}_l^{S\setminus s}$.

Combined objectives function. The combined objective of our approach as adapted from [10] is shown below. The first part ensures that the generated images should be as realistic to underwater real images as possible while the second part ensures that the corresponding patches from the input and output images should occupy similar embedding space.

$$\mathcal{L}_{\text{GAN}}(G,D,X,Y) + \mathcal{L}_{\text{PatchNCE}}(G,H,X) + \mathcal{L}_{\text{PatchNCE}}\left(G,H,Y\right) \tag{4.4}$$

4.2 Model Architecture

We use three different model architectures for this work. For the paired image-to-image translation, we use the residual network and transpose convolutional layers to build an autoencoder for image generation. The remaining two architectures are the generator and discriminator networks of our generative Adversarial network used for the unpaired image-to-image translation.

4.2.1 Autoencoder Network

We build an end-to-end encoder-decoder autoencoder system for the paired image-to-image translation (as shown in the Figure 4.2). We use ResNet34 for the autoencoder.

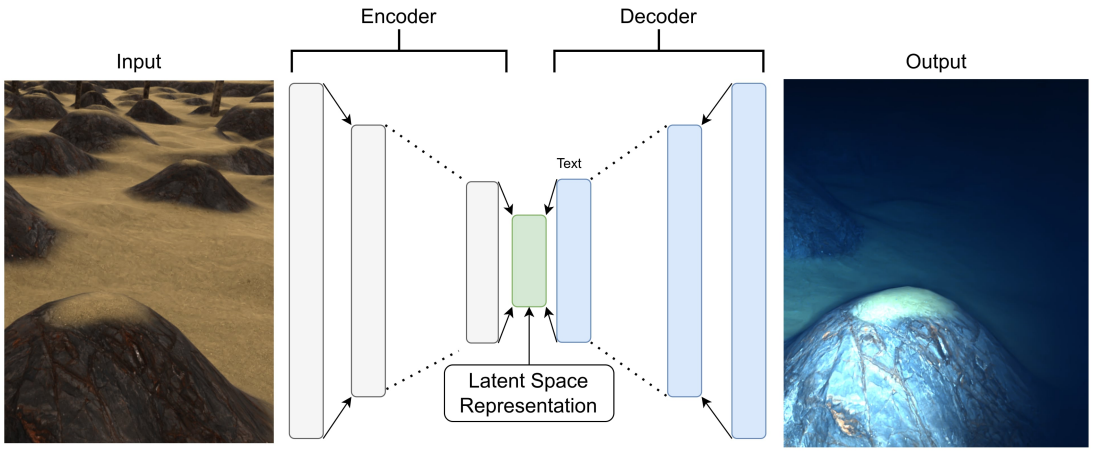


Figure 4.2: An encoder-decoder autoencoder model architecture for the paired image-to-image translation

Encoder. The ResNet34 architecture consists of blocks of 2D convolutional layers with skip connections. To be specific, the first layer of our encoder takes its 3-channel RGB image as input to a 2D convolution (Conv2D) with the square 3 x 3 kernels, stride and padding $= 1$, and output C $= 64$. This output channel goes through three blocks of the convolutional layer with each consecutive convolution block separated with batch normalization and each block ends with a ReLU activation function. The first layer maintains the output channel as 64, and the second layer increases this channel to 128 channels and consists of four Blocks similar to those discussed above. In Layer 3, the first block upsamples this channel to 256 followed by 5 identical blocks preserving the output channel. Layer 4 consists of three blocks with the first block upsampling the channel to 512 and the two remaining blocks running 3×3 kernels Conv2d with a stride of 1 and output of 512. This output is then passed through a ReLU activation function and adapted average pooling is then applied. We apply a Dropout with a probability of 0.3 on the output. Lastly, we feed this output into a fully connected linear layer that outputs our latent vector or embeddings.

Decoder. The function of the decoder is to reconstruct an output from the compressed latent vector generated by the encoder. Typically the output dimension is the same as the input dimension. In this context, we expect the reconstructed image to be a realistic underwater image with 3 channels and dimensions 64 x 64. To achieve this, we use a series of convolutional and transpose convolutional layers to upsample and downsample the input feature map to a desired output feature map. We first pass the latent vector through a linear

layer, a ReLU activation, and an Unflatten layer to process the channel to be suitable as a feature input into the first transpose convolutional layer. Next, we run a series of four blocks with varying output feature maps with each block having a transpose convolutional then a ReLU, and another convolutional layer followed by another ReLU. The output of the decoder network is the same dimension as our input and is used in computing the MSE loss along with the expected ground truth.

4.2.2 Adversarial Network

We use a generative Adversarial network with its generator and discriminator architecture for the unpaired image translation. Adapted our generator and discriminator architecture from those in [10] which is the same as that of CycleGAN. The core of the generator and discriminator architecture is discussed in the following subsections

Generator. The generator architecture is remodeled by Johnson *et al.* [46]. The model contains a series of normal fractional strided convolutions and several residual blocks. In this thesis work, we use 9 blocks of residual layers in our experiment. All non-residual layers are followed by Batch normalization and relu activation function with the exception of the output layer.

Discriminator. The discriminator of a GAN network takes images and tries to classify whether the images are real or fake. Specifically, in our case, if the images are real underwater images or refined realistic-looking underwater images. We use the PatchGAN developed by pix2pix [51] for our discriminator architecture. The idea behind PatchGAN is to restrict our focus to smaller Patches that capture high frequencies structure in an image. PatchGAN only penalizes structure at the scale of small patches. A binary classification of real or fake is performed on all n by n patches and all outputs are averaged to produce the overall discriminator output.

4.3 Data Processing

The data used in this project is crucial to obtain good results as shown in Chapter 5. For this Master's thesis, we use three major collections of datasets. Depending on the image translation technique, we preprocessed these datasets differently. The three main categories of the dataset are:

I. Vision Autonomous Robots for Ocean Sustainability (VAROS) dataset

II. Underwater Image Enhancement Benchmark (UIEB) dataset

III. Our custom underwater dataset generated from a YouTube video sequence

These datasets have been discussed in depth in Chapter 2, therefore in this chapter, we focus on the data processing and the data transformations. The dataset setup during the training of paired and unpaired image translation is shown in Figure 4.3.

Paired image translation. For the experiment involving paired image-to-image translation with encoder-decoder autoencoder, we use a subset of the VAROS dataset. We extracted 751 uniform lighting and their paired underwater equivalent. This sequence of images corresponds to the sequence 0 to 755 in the VAROS data. The idea behind our autoencoder method is that by giving an input batch of uniform lighting images, we hope to learn a model that produces realistic underwater images while preserving the input content. We achieve this by optimizing our autoencoder model with the MSE loss between the generated

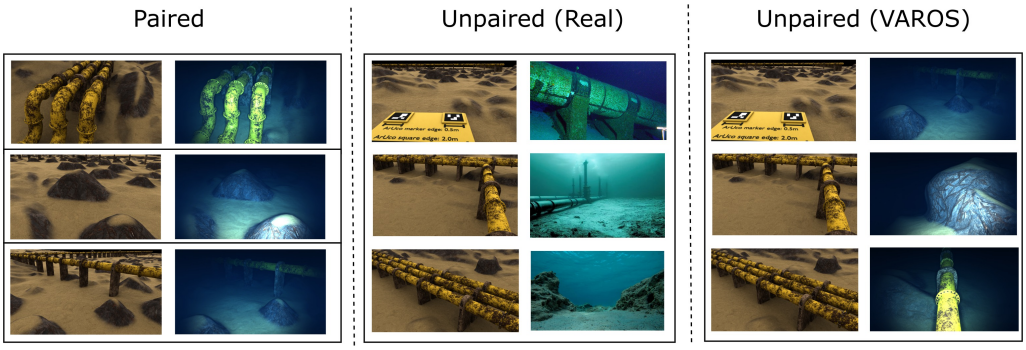


Figure 4.3: Paired training data (left) consists of training examples $\{x_i, y_i\}$, where the correspondence between x_i and y_i exists from folders B and A in the VAROS synthetic underwater dataset respectively. We consider unpaired training data (middle), consisting of a source set $\{x_i\}$ $(x_i \in X)$ and a target set $\{y_j\}$ $(y_j \in Y)$ sampled from real underwater images, with no information provided as to which x_i matches which y_j. Unpaired training example by using the synthetic input x_i for $i = 1$ to K from VAROS and a target set y_j for $j = K - N$ to ensure no correspondence.

images and the expected ground truth, in this case, the equivalent underwater folder from the VAROS data. We formulated a data loader that transforms the PIL images to tensor and resizes the images to 64 x 64. At each iteration of the training procedure, the Dataloader returns a batch of uniform lighting images - input to our model - and the corresponding underwater ground truth images for MSE loss computation. We use the same arrangement of data for training the pix2pix model.

Unpaired image translation. For this setting, our objectives as somewhat more complicated and so are our datasets. The overarching goal of unpaired image-to-image translation, and by extension this Master's thesis, is that given a set of uniform lighting synthetic datasets, we aim to generate realistic-looking underwater images by conditioning a GAN Refiner model on real underwater images. Since the real-world underwater images can be collected from any random underwater environment without any correspondence both in terms of content and structure to the input uniform lighting images, we categorize these as an unpaired image-to-image translation problem. We created three groups of unpaired datasets. The first was extracted from the VAROS synthetic datasets platform. We select 1090 uniform lighting images corresponding to sequences 1011 to 2100. For the realistic underwater images, instead of selecting a correspondence sequence as in the case of paired image translation, we rather select 1011 images corresponding to sequence 0000 to 1011. For the next group, We retain the 1090 uniform lighting VAROS synthetic dataset, however, we conditioned this on the UIEB [30] real underwater images consisting of 890 image samples of a varying underwater scene. Lastly, we developed our custom dataset by extracting sequences from the video [53] mixed with randomly selected freely available internet images. Our custom data set consists of 511 images. A comprehensive experiment using different models on these three groups of data sets is presented in the Chapter 5.

4.4 Model Evaluation

Evaluating the realism of the output-generated images in autoencoder and GAN is an open and difficult problem. Aside from assessing the visual quality of the generated data by subjective human ever evaluation, we also compute the MSE and SSIM between the realistic

generated output and the expected underwater ground truth. We are aware that these metrics do not capture the joint distribution of the results, nevertheless, we adapted them to provide some useful insight on the output of our network. Furthermore, we use the popular FID metric which computes the divergence between the estimated distribution of real and generated images. Ideally, we want the realistic images to occupy similar distribution of real underwater images regardless of the feature space.

Chapter 5

Experimentation and Results

This chapter covers some of the key experiments in this master's thesis project and carefully links each experiment to the overall goal of this project. We started by presenting the experiment setup which includes datasets and training information for both the paired and unpaired image-to-image translation experiments. Before that present a detailed description of the results obtained while also performing some quantitative evaluation based on the evaluation metrics presented in the previous chapter. Lastly, we compare these results with established benchmarks or baseline models. A comprehensive analysis and discussion of the results obtained in this chapter are introduced in Chapter 6.

5.1 Experimental Setup

The experiment in this report is divided into two major categories - paired and unpaired image translation. For the first category, we build an end-to-end autoencoder model and compare our results with the state-of-the-art in paired image translation pix2pix [51]. Similarly, for the unpaired image translation, we adapted [10] to account for depth information and compare our results with some established baselines [8, 9, 50]. We presented our code for the implementation of the SimGAN [8] and WaterGAN [50] in our Github repository. As for the CycleGAN [9], the authors have provided a rich implementation on their page here. The remainder of this subsection introduces the reader to the dataset, network architecture, training details, and speed of each category of the experiments. Some specific details on hyperparameters have been omitted from this section for the sake of brevity and can be found in the papers.

5.1.1 Paired image to image translation

Since the data setup and architecture used for the paired image translation are inherently different from the unpaired image translation, we have carefully highlighted this setup in this section. It is important to note that the data setup presented here is only applicable for the training of our autoencoder and pix2pix model.

Dataset. The VAROS dataset provides a good collection of uniform lighting and the underwater equivalent of the same scene as shown in Figure 2.3. The VAROS folder structure is shown in 2.5.1. Folder A contains the underwater sequence and is used as a label. Folder B contains the corresponding uniform lighting sequences which are the input of our model. The C and D contain the surface normal and depth map respectively. We focus on just the first two folders for this experiment. We select 751 training examples from folders A and B for

training both our autoencoder and pix2pix model. We evaluate all our models irrespective of their category on the set of carefully selected images containing some edge cases. The training set is selected from the VAROS environment and placed in two separate folders. The uniform lighting and underwater sequence are sufficient for training our autoencoder model with the custom Pytorch Dataloader. However, for the pix2pix, further processing is needed. The pix2pix code as provided by the authors needs the input pair to be stacked side by side in a single folder (Figure 5.1). To achieve this, first, we need to ensure corresponding image sequences have the same name. For example, we rename sequence seq01_veh0_camM0_A-00000001 and sequence seq01_veh0_camM0_B-00000001 to 00000001 and 00000001 in folders A and B respectively. Next, we use the script provided by the authors to concatenate this sequence horizontally to form a new folder BA with 751 images. Examples of these new images are shown in Figure 5.1

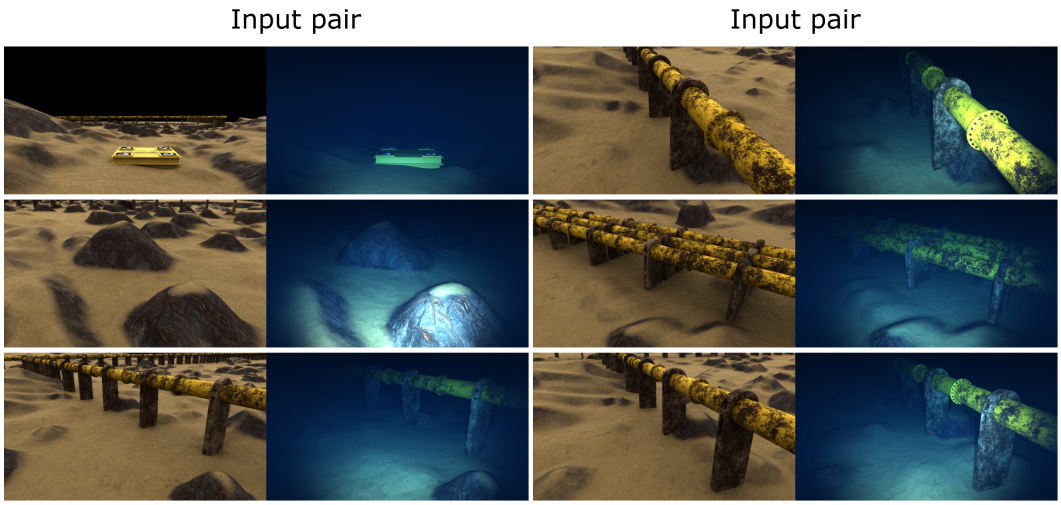


Figure 5.1: Paired training data (left) consists of training examples $\{x_i, y_i\}$, where the correspondence between x_i and y_i exists from folders B and A in the VAROS synthetic underwater dataset respectively, for training the autoencoder and pix2pix model

Network. For the paired image translation, we explored two main model architectures. Our autoencoder network and the pix2pix network. For this work, we use the encoder-decoder architecture which we have explained in detail in Section 2.4.4, and compare this with the baseline pix2pix [51]. The autoencoder network is composed of convolutional and transpose convolutional networks and most importantly the encoder or feature extraction head is made up of the ResNet34 architecture. The pix2pix model is composed of two networks, the Unet256 is used for image generation, and the PatchGAN discriminator architecture is used for enforcing underwater realism in the model output. We refer the reader to the pix2pix [51] and code by the authors for a detailed explanation of the model architecture and implementation.

Training details. This section is intended to ensure that experiments in this Master's thesis are reproducible. We train our autoencoder model with the following hyperparameters learning rate (lr) $1e^{-3}$, we apply the Adam optimizer with a weight decay of $5e^{-5}$. These values were selected based on empiricism and had no theoretical backing. In addition, we use the MSE loss as the optimization criterion. We trained our autoencoder network for 500 epochs on the Calypso RTX 3090 GPU provided by Alteia. We employ two techniques to visualize our training loss and image generation. First, we use the Jupyter Notebook environment to view the running loss of images generated at every 50 epochs as shown in

Figure 5.2. This provides easy access to training metrics and generated results. Second, for each time tracking of the training procedure, we log the training metric and generated an image on the WandB platform. On GPU, our autoencoder model trains for an average of 40 seconds per epoch and 12 minutes for epoch on CPU. The input to our model is resized to a 256 x 256 RGB square image. This input is converted to tensors using the Pytorch transform function. During inference, we apply the trained model on the test dataset and expect realistic underwater images with 256 x 256 square dimensions.

5.1.2 Unpaired image to image translation

The experimental setup of the unpaired image translation is significantly different from the paired counterpart. Unpaired image translation can, in theory, learn a mapping that takes images from one domain to another given examples of unpaired images in both domains. In our case, this mapping aims to translate synthetic uniform lighting images into realistic underwater images. The aforementioned goal introduces the need to carefully select unaligned datasets, network architecture, and a stable training procedure.

Dataset. To achieve unpaired image-to-image translation, we structure our dataset to encapsulate the two domains we aim to capture in this research. We successfully created an unaligned pair with the VAROS dataset. This unpaired sequence of the VAROS dataset underpins the ability of our model to learn a mapping **F** that effectively translates images from one domain to another in the absence of paired training examples. The corresponding depth map in folder D is used in our CoDe method. Specifically, we selected sequences A-00000000.png to A-00001010.png from VAROS folder A as the synthetic underwater images and sequences B-00001011.png to B-00002100.png in VAROS folder B as the unaligned uniform lighting synthetic training examples. Notice that we have intentionally selected uneven numbers of images for each domain. For example, we selected 1011 images for the synthetic underwater image while 1090 were selected for the uniform lighting. we did this to buttress the idea that this is an unpaired image translation dataset and as such there is no explicit need for the number of training images in each folder to be the same. Next, to ascertain the robustness of our model, we introduced two additional real underwater datasets. First, we replace the 1011 synthetic underwater images from VAROS with the 890 real underwater images from the UIEB dataset [30]. second, we generate a total of 511 real underwater images from a video sequence [53] and freely available real underwater images on the internet. We call the latter our custom underwater dataset. Note, in both new datasets the uniform synthetic images remain the same, our only replacement is the underwater images in each scenario.

Network. We adapted the architecture provided in [10] for our unpaired image translation and made modifications to the code provided by the authors to account for the extra depth channel in our VAROS dataset. Our goal is to use the depth information to force our model to learn underwater light attenuation. A detailed explanation of our network has been provided in Chapter 4. On a high level, the network consists of a GAN architecture with a PatchGAN as the discriminator and a contrastive loss as a substitute for the cycle-consistent loss used in CycleGAN. For easy comparison with the results of other baselines in the unpaired image translation, we have named our modified algorithm CoDe (contrastive+depth). In the next subsection, we present the results obtained using this architecture and compare them with those from baselines [8, 9, 50].

Training details. The hyperparameters of the CoDe network are the same as those used in CycleGAN [9] except that the loss function is replaced with contrastive loss. We train the CoDe using 256 x 256 4-channel RGBD image with 9 residual blocks, PatchGAN discriminator, least square GAN loss, Adam optimizers, lr $2e^{-3}$, and a batch size of 8. We train for

200 epochs, which is similar to CycleGAN for a fair comparison. For the calculation of the patch based on contrastive loss, the authors in [10] extracted features from 5 layers. The layers correspond to receptive field sizes of 1x1, 9x9, 15x15, 35x35, and 99x99. The authors further sample 256 random locations for each layer's feature and use a 2-layer MLP as a projection head for obtaining the final 256 dimension embeddings. The momentum value and temperature were 0.999 and 0.7 respectively. The size of the memory bank is 16384 per layer. For the other two baselines [8, 50], the training parameters were slightly less complicated. For the WaterGAN, since we only implemented the attenuation equation from the paper using the depth map, there was little need for training any model, rather we adjust the wavelength parameters, η, from the three color RGB channels to the following values 0.00008, 0.000015 and 0.01 respectively to get the results shown in Figure 5.7. The training of SimGAN is straightforward, we follow the standard GAN training procedure detailed in the paper [8]. As presented in the paper, we use 4 residual blocks with skip connections with input and output channels of 64 in the Refiner network while the discriminator consists of 5 blocks of convolutional layers with a Maxpool layer after the first two blocks. we use a weight decay of $5e^{-5}$, lr of $1e^{-3}$, and SDG optimizer for both the Discriminator and Refiner networks and the λ of 0.001. We adopted the image history buffer with a capacity of 10 x batch size in training the discriminator. In this context the batch size is 32, therefore our image history buffer can store 320 image tensors. the training procedure is the same as described in the paper, we pre-train the Refiner network for 500 steps and the discriminator for 200 steps, then we train the combined networks for 10,000 steps updating the Refiner twice and the discriminator once at each step.

5.2 Results

We begin by investigating the training convergence in both our autoencoder and GAN network. We also compare our approach with some recent methods for unpaired image translation using some established benchmarks. We then performed an ablation study to understand the relevance of depth maps in the quality of the generated images. Lastly, we explore the explainability of our model by performing additional experiments to extract embeddings from different layers. For simplicity, during visualization, we refer to our Method as CoDe (Contrastive+Depth)

5.2.1 Training Convergence

In Figure 5.2 we show a plot of the autoencoder training loss for 500 epochs with a learning rate (lr) of 0.002. We observe that using a smaller lr facilitates early convergence of our model. The generated images using this model are presented in Figure 5.3. Notice how the results are blurry, this is because our encoder-decoder is trained on a traditional metric such as the MSE that doesn't capture high-level features leading to blurry output. For the GAN training, we try to visualize both the generator and discriminator losses at every stage of the training process. Unfortunately, the training losses of a GAN model do not provide much insight into the model's performance during training, and our model is no exception.

5.2.2 Evaluation Metric

As we have reiterated severally in this report, the evaluation of the realism of the output generated by GAN is tricky and an open problem. However, since the expected outputs from

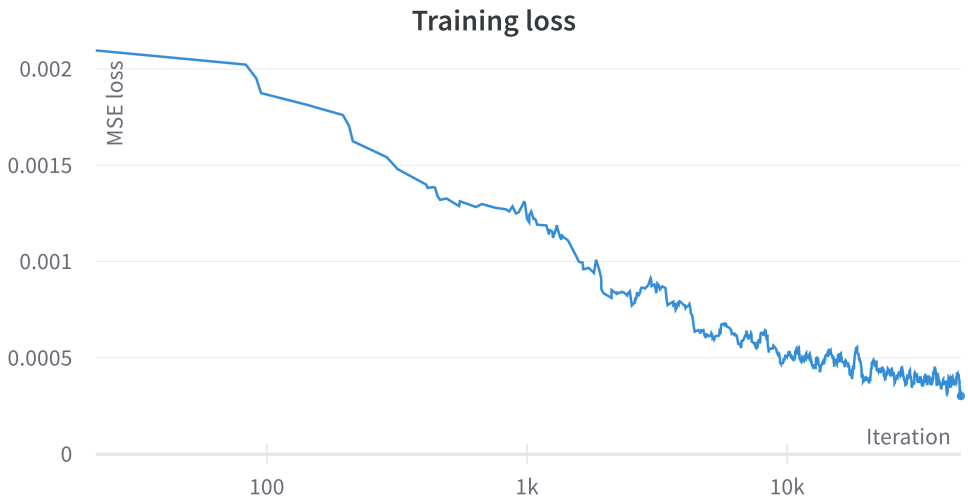


Figure 5.2: Autoencoder model training on the paired images from the VAROS synthetic. Mean Square Error (MSE) loss computed for the entire 500 epochs

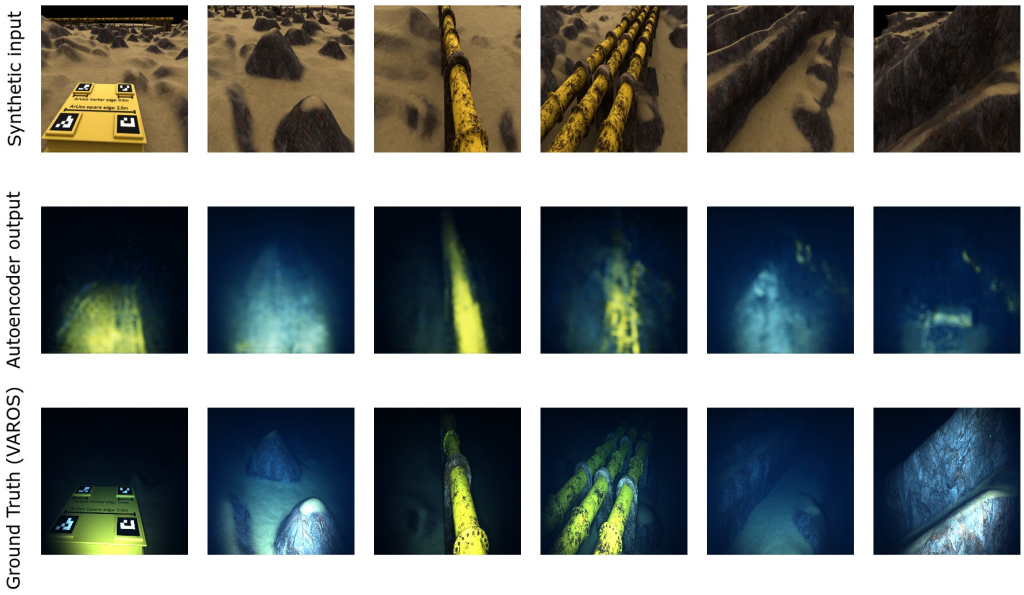


Figure 5.3: Comparison of the generated results from our paired autoencoder model with the VAROS underwater ground truth.

our method are realistic underwater images and we have a base idea of what these images should look like as well as the expected ground truth images from the VAROS dataset, we have carefully employed a series of qualitative and quantitative metrics to evaluate our results.

Fréchet inception distance (FID) [12]. This metric measures the similarity between two image datasets, it has been shown to correspond to human subjective analysis of the visual quality of samples from GANs. FID computes the distance between two Gaussians fitted to feature representation of the inception network. Throughout this master's thesis, unless otherwise stated, the FID is computed between the generated output and the ground truth from folder A in the VAROS dataset. We use the Pytorch FID code developed by [54] to compute all displayed FID scores in this report.

Structural Similarity Index Metric (SSIM) [11]. The SSIM compares the structural information between two images rather than just the pixel-level values. The SSIM index ranges from -1 to 1 with -1 indicating complete dissimilarity and 1 indicating perfect similarity. A value of 0 indicates that the images are uncorrelated. We compute the SSIM on all 190 images in our test set using their corresponding ground truth from the VAROS environment and the generated results. The FID and SSIM scores are shown in Table 5.1

Underwater Colour Distribution. This is a qualitative metric that analyzes the color distribution of our underwater realistic images. We formulated this metric to intuitively provide a visual cue on the generated results. The idea behind this metric is that since our application is related to the underwater environment, backscatter, attenuation and refraction in underwater media will have varying effects on each color channel. Therefore by analyzing the distribution of the ground truth image color band as shown in Figure 5.4 and comparing this with the realistic underwater generated images, we can reach a good enough conclusion on the realism of the generated image in an underwater context. The code used for this analysis is provided in the Appendix A.2.

Feature Space of Both Classes. This is also another qualitative metric. In this metric, we simply perform a principal component analysis (PCA) on both the uniform lighting images and underwater images from the VAROS dataset on a 2D feature space to see the space both categories of data occupied and if at all both classes are linearly separable. Understanding the space that each class category occupies in the feature space is useful during inference, as we expect the realistic generated underwater image to occupy a similar domain as the ground truth from the VAROS data. A scatter plot showing the 2D points of 200 data samples of the uniform input and underwater ground truth from the VAROS platform is presented in Figure 5.5. In addition to PCA, we performed T- distributed Stochastic neighbor embedding (TNSE) (Figure 5.6) and compare our plot with PCA in Chapter 1. See the appendix for the good snippets used for this analysis in Appendix A.3

5.2.3 Baselines

Our baselines are divided into two categories (I.) Trained on **paired** data and (II.) Trained on **unpaired** data

Pix2pix [51] The model is trained on paired data and is compared to our autoencoder model. Also, to check the robustness of our CoDe model we compare our output with those of pix2pix.

SimGAN [8] This GAN model aims to learn a Refiner, that given a set of synthetic images conditioned on real images, produces realistic underwater images with similar distribution as real images while preserving the annotations of the synthetic image.

WaterGAN [50] We have implemented only a subset of the overall WaterGAN model. We

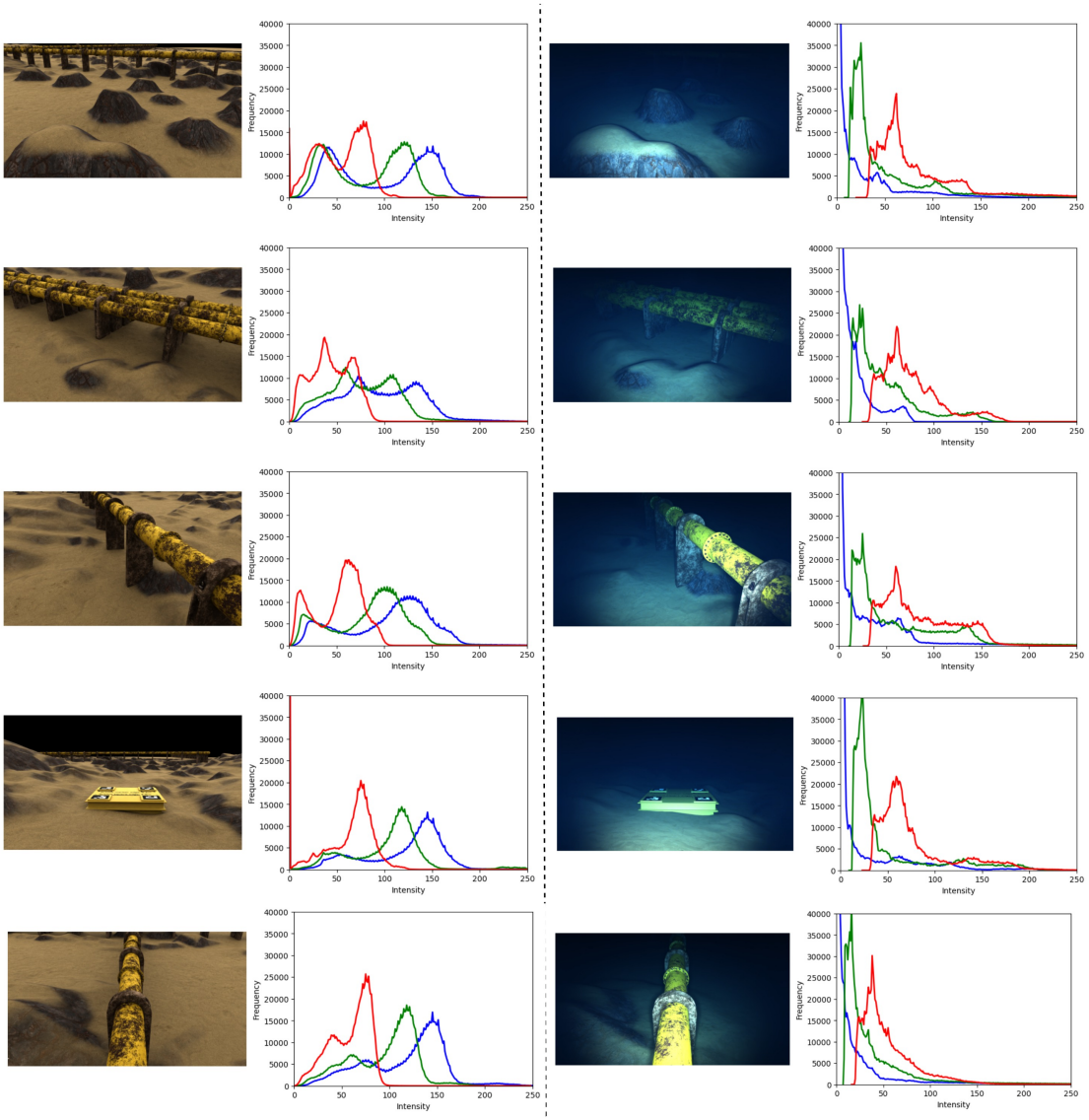


Figure 5.4: Underwater images RGB color distribution. Synthetic uniform lighting input (1st column), synthetic uniform lighting RGB histogram distribution (2nd column), VAROS realistic underwater images ground truth (3rd column), VAROS realistic underwater images ground truth RGB histogram distribution (4th column)

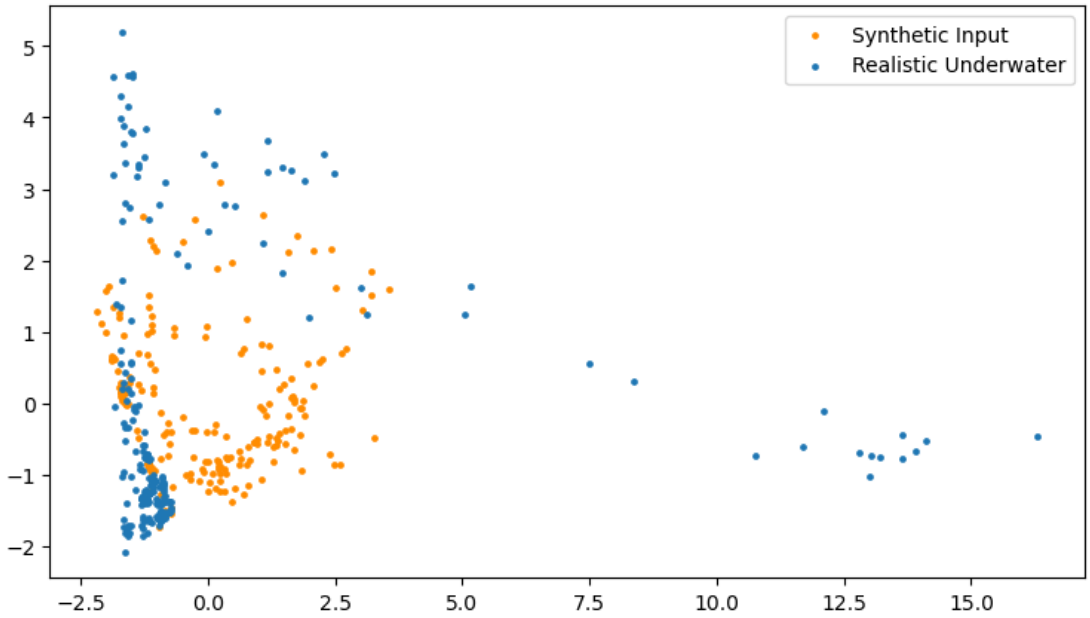


Figure 5.5: PCA Feature Space of Both Classes. A scatter plot showing the synthetic uniform lighting image (orange) and realistic underwater images (blue) position in 2D space

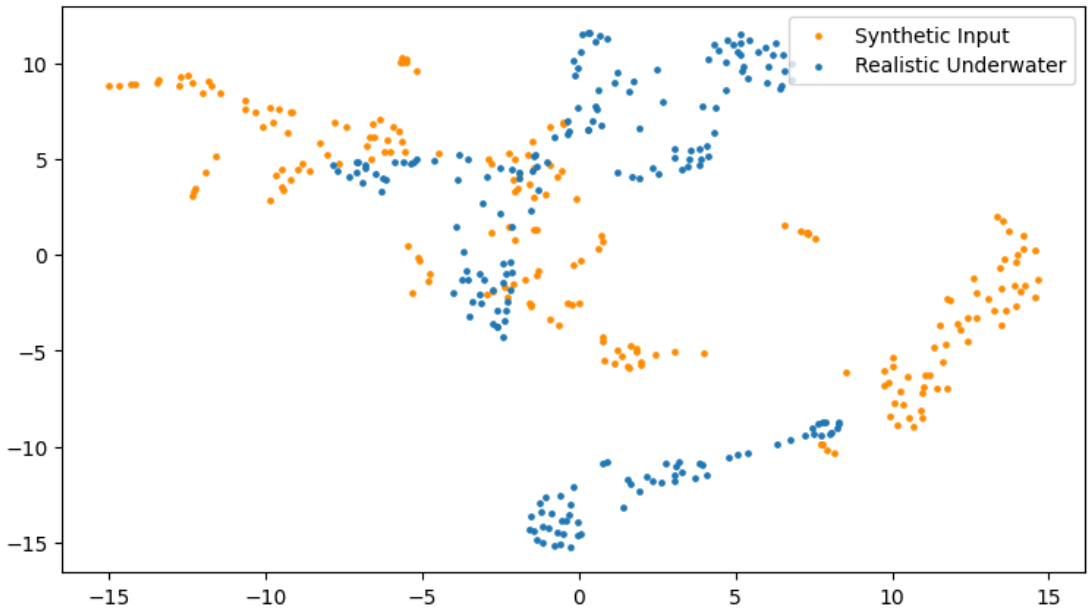


Figure 5.6: TNSE Feature Space of Both Classes. A scatter plot showing the synthetic uniform lighting image (orange) and realistic underwater images (blue) position in 2D space

use the depth maps to attenuate the synthetic image using the equation $G_1 = I_{air}e^{-\eta(\lambda)r_c}$. The outputs of this method are shown in Figure 5.7.

Gatys [44] This method performs style transfer on an image pair, rather than learning a model for cross-domain translation. [44] takes two input images one with the style in our case the real underwater images and the other with the content and creates a new image that maps the style of the style image to the content of the content image.

CycleGAN [9] Our approach is similar to the CycleGAN with only a slight difference in the loss function. Our approach replaces the cycle consistent loss with contrastive loss, without the need for an extra discriminator and generator network inadvertently improving speed and memory. For a fair comparison, we train all baselines for similar epochs and use similar parameters where possible.

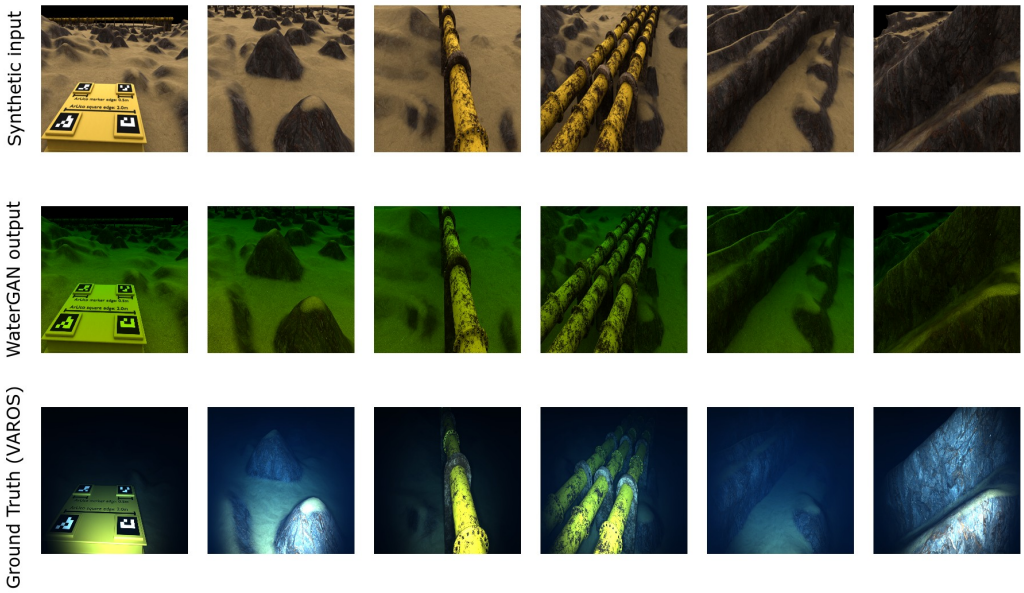


Figure 5.7: Comparison of the generated results from the **WaterGAN** model using just the attenuation equation with the VAROS underwater ground truth.

5.2.4 Comparison with Baselines

Figure 5.3 shows the result obtained using our autoencoder-based image generation. We performed the comparison separately as both models were trained using paired data. The pix2pix trumps the naive autoencoder-based image generation. This can be attributed to the fact that pix2pix uses the PatchGAN discriminator to enforce high-level features in an image while the autoencoder uses traditional metrics like the MSE loss which focuses on low-level pixel similarity while ignoring overall structural information and thus leading to blurry results. We further provided the quantitative comparison with the ground truth by computing the SSIM and FID scores shown in Table 5.1.

On the other hand, for the unpaired image-to-image translation, we compare the result from our CoDe model against several baselines in Figure 5.9. From Figure 5.9, CycleGAN and CoDe (ours) were able to achieve compelling results as compared to the SimGAN, WaterGAN

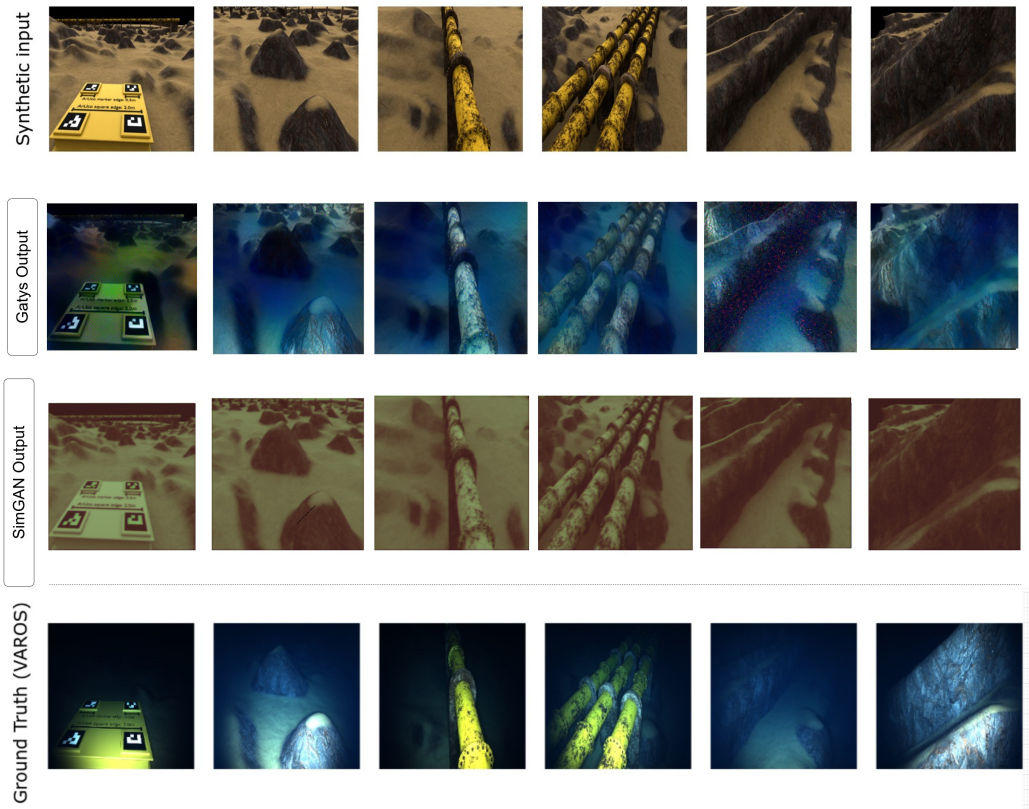


Figure 5.8: Comparison of the generated results from the **Gaty's** style transfer method and **SimGAN** with the VAROS underwater ground truth.

Table 5.1: Comparison with baselines. We compare our methods with other baselines on the VAROS underwater datasets with the FID and SSIM evaluation metrics. CoDe denotes our model trained with depth information. CUT is similar to our model but without depth information. We show FID, a measure of image quality [12] (lower is better) and the SSIM (closer to 1 is desired). We show results on inference from a random 6 samples of the VAROS underwater lighting images and the entire 190 test images. Based on the FID quantitative measures on all 190 test images, CoDE produces higher quality and realistic underwater image generations with a lower footprint in terms of training speed and GPU memory usage as compared to the CycleGAN in second place.

	Samples (6)		Samples (190)	
Method	SSIM↑	FID↓	SSIM↑	FID↓
Pix2Pix	0.66	206	0.68	128
Autoencoder	0.70	314	0.67	268
SimGAN	0.36	341	0.41	262
WaterGAN	0.31	257	0.29	185
CycleGAN	0.61	241	0.67	108
CUT	0.62	299	0.69	132
CoDe	**0.57**	**278**	**0.66**	**106**

and Gatys. we acknowledge the generative capability of the CycleGAN model, however, our model was able to achieve comparable results despite being more memory efficient and faster than the CycleGAN during training. For example, our CoDe model has a total of dash 14.7M parameters. In contrast, CycleGAN has a total of dash 28.3M parameters because the CycleGAN has a double generator and discriminator architecture making our approach two times more memory efficient and 30% faster than a CycleGAN with the same architecture when trained on NVIDIA RTX 3090. A careful look at the FID and SSIM scores comparing our CoDe method with other baselines underpin our earlier assertions that our model despite having lesser parameters achieved the best FID score on the unpaired image translation task. Although the FID of the pix2pix is higher for just 6 samples, to an extent, we can see that this is because the pix2pix is trained on the paired examples and the 6 samples were part of the training examples. For the 190 samples, the pix2pix achieved lower results than our CoDe method. We surmise that the large test samples contain some edge cases which are absent in the pix2pix training. In both cases, our CoDe method outperforms the CycleGAN, CUT, SimGAN, Gatys, Autoencoder, and WaterGAN baseline.

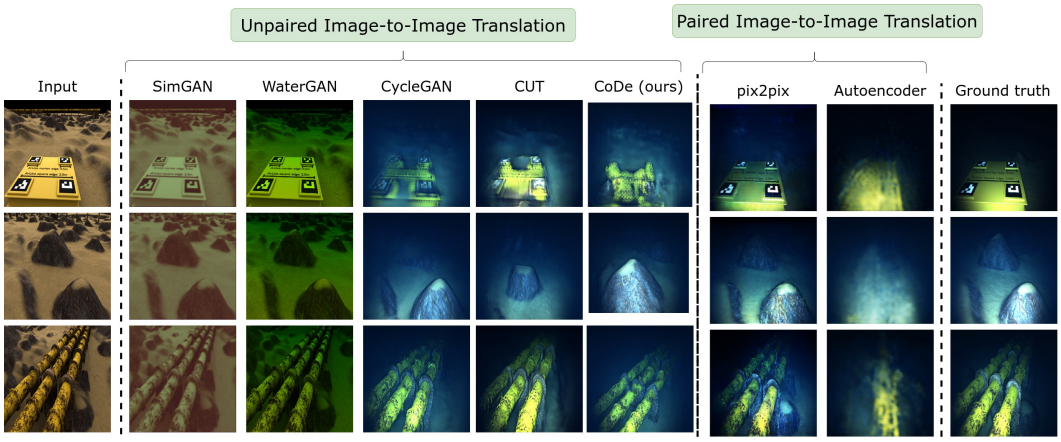


Figure 5.9: Comparison with baselines. We compare the images generated by our CoDe on the VAROS underwater datasets with other baselines for paired and unpaired image translation. CoDe denotes our model trained with depth information. CUT is similar to our model but without depth information. We show results generated using pix2pix and autoencoder paired image translation models.

5.2.5 Ablation Study

We discovered that in the underwater scenario, the depth map is relevant in capturing the water attenuation and illumination. we compare the generated output using depth and no-depth information. By default our CoDe model accounts for depth information to produce the result in the earlier sections. This extra depth encourages our model to learn subtle attenuation properties in underwater images.

In Figure 5.11 we show the results using depth and without depth which are represented by the CoDe and CUT method respectively. The FID score is shown in Table 5.1 - The vanilla CUT has no depth in the input while our CoDe model is adapted to accepting depth as input during training and inference.

Replacing the VAROS Synthetic Underwater Data. We also explored other variants of real underwater images to condition our code model rather than using just the VAROS underwater data. It is important to note that in all the cases the synthetic uniform light-

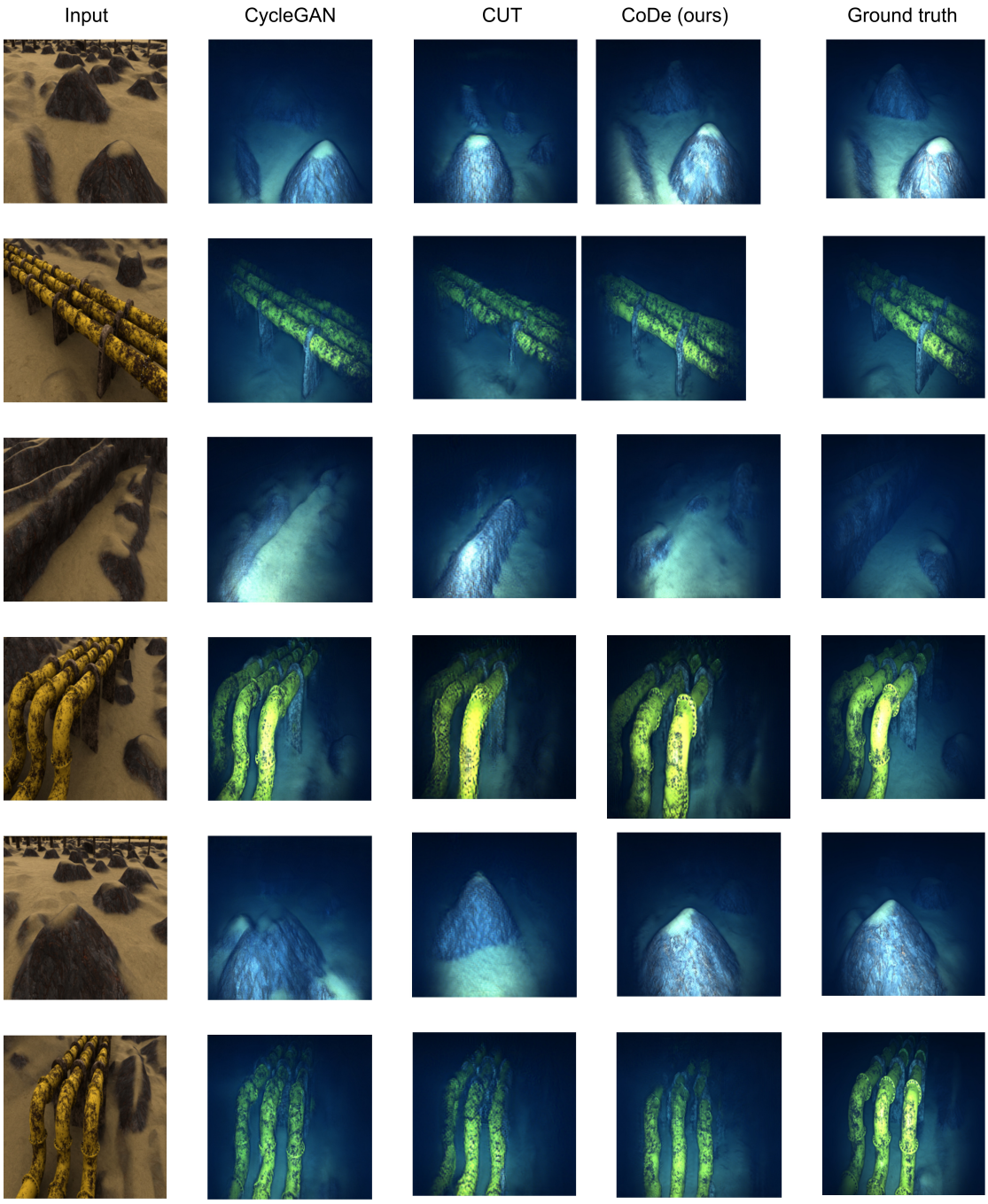


Figure 5.10: Comparison with baselines (Unpaired image translation). We compare the images generated by our CoDe on the VAROS underwater datasets with other baselines for unpaired image translation. CoDe denotes our model trained with depth information. CUT is similar to our model but without depth information.

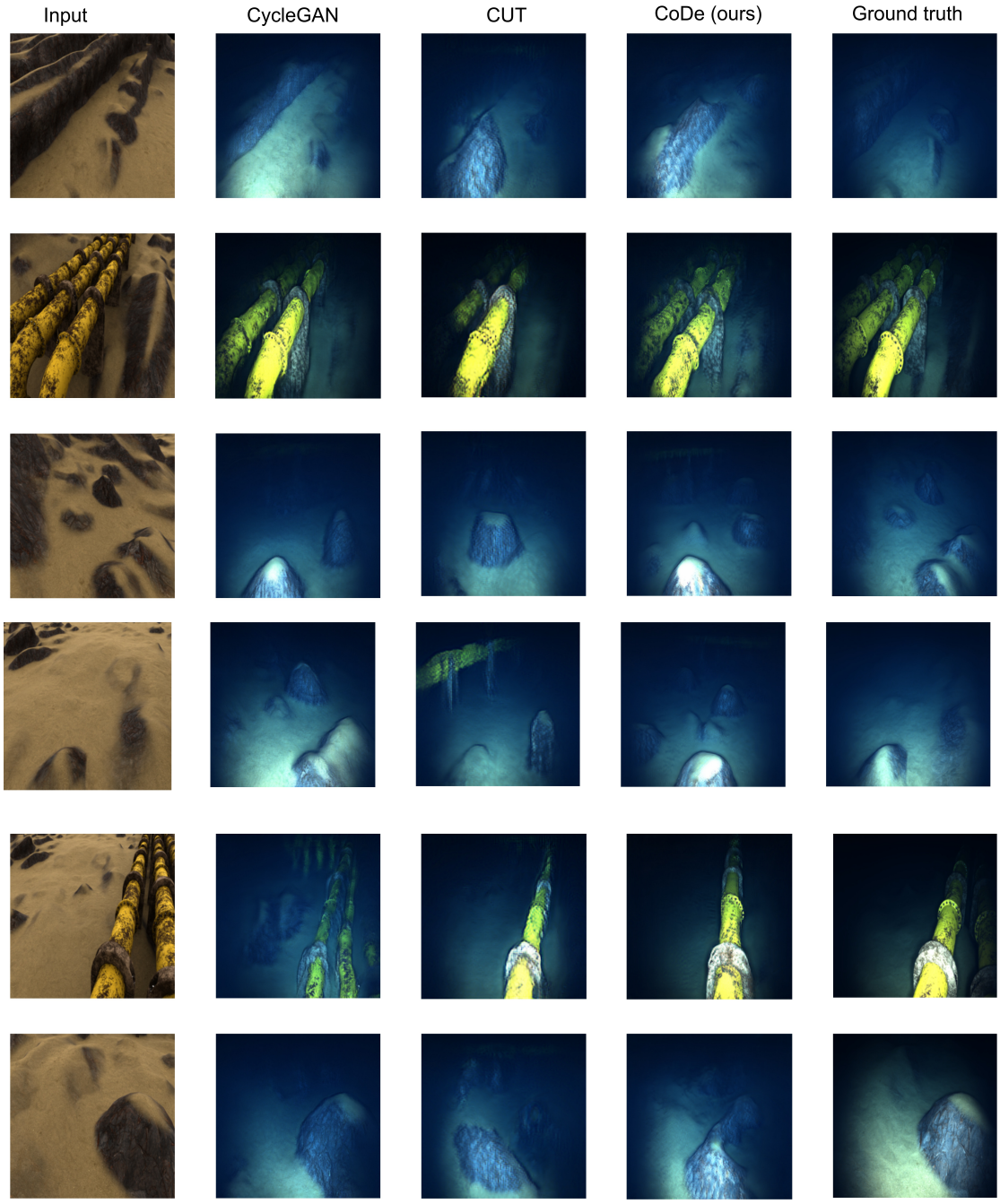


Figure 5.11: Comparison with baselines (Unpaired image translation). We compare the images generated by our CoDe on the VAROS underwater datasets with other baselines for unpaired image translation. CoDe denotes our model trained with depth information. CUT is similar to our model but without depth information.

ing input is still from the VAROS dataset. Figure 5.12 shows the results we obtained when we conditioned on real underwater images from the UIEB [30] dataset and our custom underwater datasets [53]. At first glance, we noticed some bright coloration on the results generated when trained on our custom data, however, after rigorous analysis, we conclude that the dataset has a lot of images with sharp sunlight reflections. these forces are models to learn the bright spot for improving the realism of synthetic images erroneously assuming that the bright spots are essential components of real underwater images. To solve this, we carefully removed all images with bright spots from our custom data and retrained the model without bright spots. A few bright spots images are shown in Figure 5.13. We have presented the generated result without bright light spots as the fourth row of Figure 5.12.

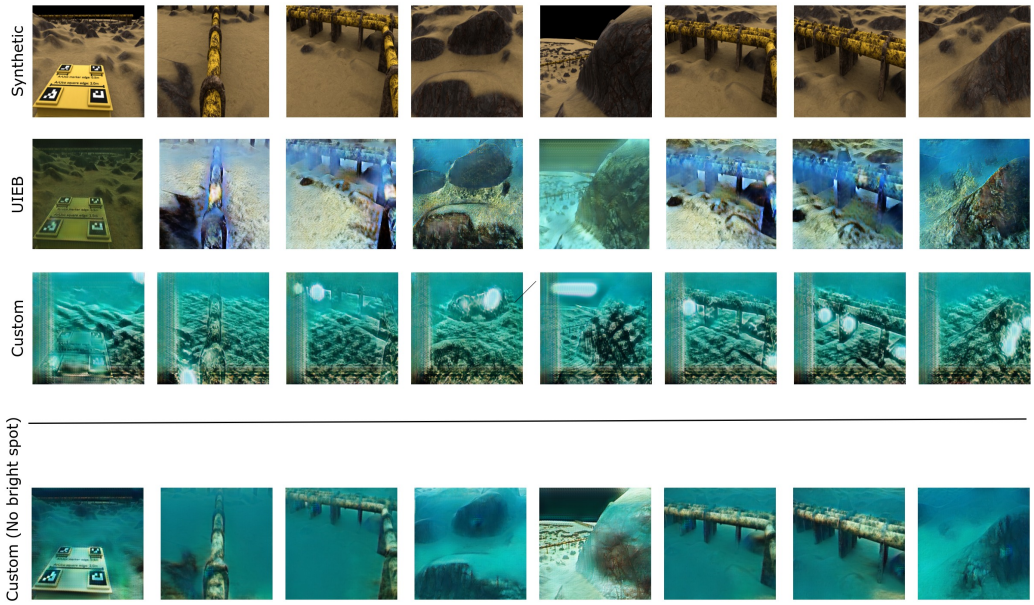


Figure 5.12: The result obtained training our model using the synthetic input from the VAROS underwater dataset conditioned on real images from the UIEB and our custom dataset.

5.2.6 Additional Experiments

In addition to the above experiment and comparison to the baseline, we further explore some inherent characteristics of our model to enhance explainability.

Different layers activations. We visualize the output from different layers of our model and display these results in Figure 5.15. We hope that understanding the learned features of each layer would provide us with a better idea of our model performance and training. To achieve this, we visualize the weights and the activated outputs from layers 4, 8, 19, 22, 43 and 63 of our autoencoder model. The filter sizes are all 7 by 7. The weight provides useful insights into the learned features of each layer. For example, Layer 43 plays a pertinent role in extracting the light attenuation of underwater images. We have provided a code snippet for this experiment in the Appendix.

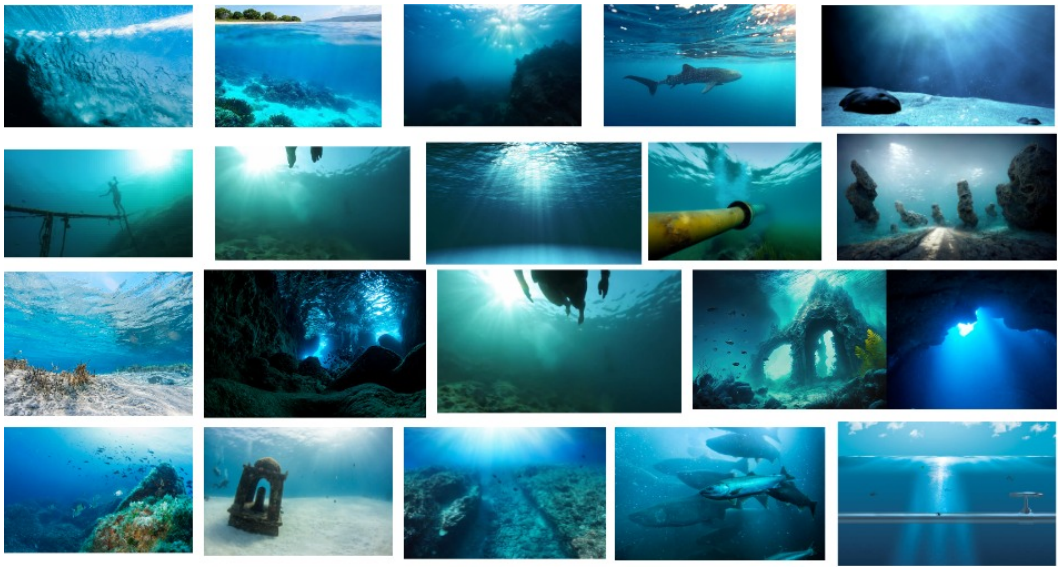


Figure 5.13: Bright spot images from sunlight cleaned from our custom dataset to improve the accuracy of the generated images.

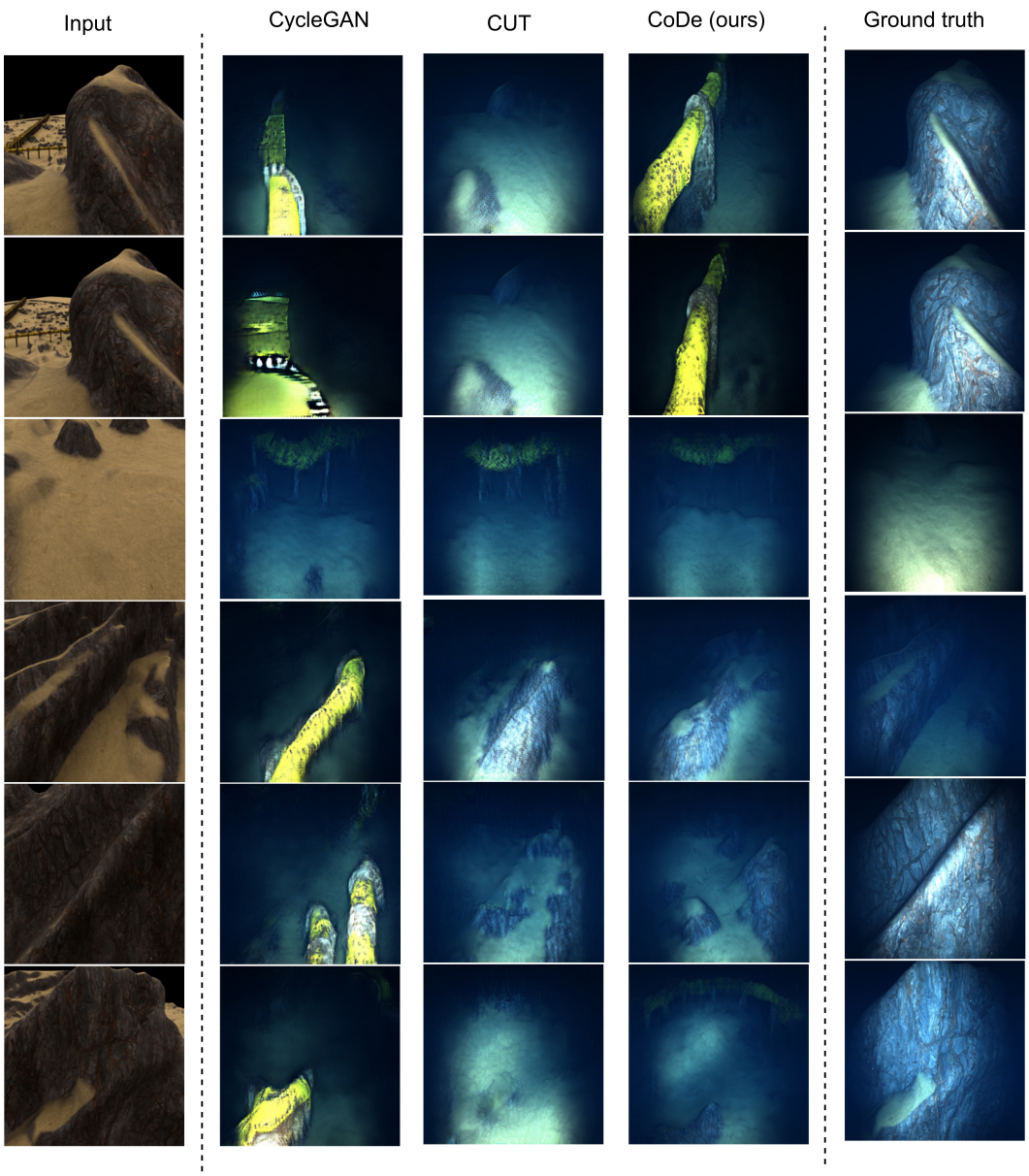


Figure 5.14: Typical failure cases of our method and other unpaired image translation techniques or difficult inputs. CycleGAN and CoDe model generating images with color and texture of pipelines from a uniform lighting input without pipelines

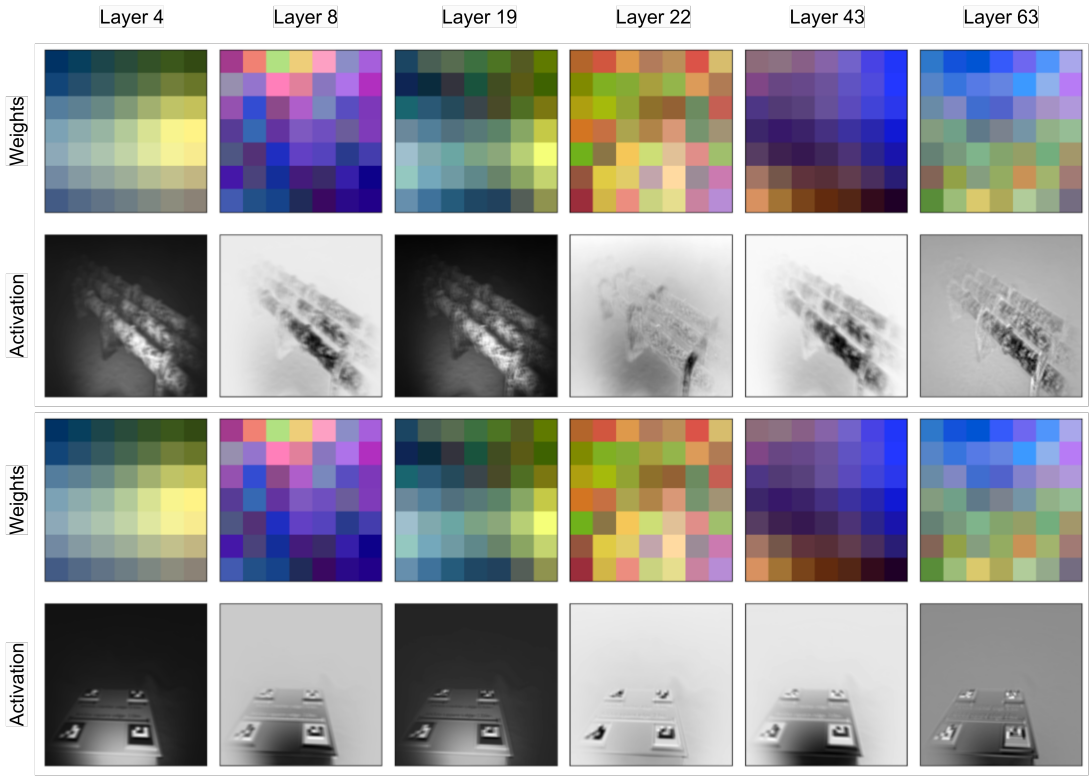


Figure 5.15: We examine the weights and activated outputs of specific layers (4, 8, 19, 22, 43, and 63) in our autoencoder model. The filters in these layers have a size of 7 by 7. Analyzing the weights allows us to gain valuable insights into the learned features of each layer.

Chapter 6

Discussion

In this chapter, our goal is to provide a comprehensive analysis and discussion of the results from Chapter 5. Given that this Master's thesis doubles as a detailed overview of the application of some state-of-the-art techniques on image translation in the underwater environment specifically using the VAROS datasets, we have not focused our discussion on the CoDe method alone, rather we present the results and limitation of other baselines as well. For ease of navigating this chapter and to ensure we touch upon crucial aspects, we structured this chapter to model a similar outline as in Chapter 5.

6.1 Dataset

The dataset setup forms a significant part of this work. This is evident in the core aim of this thesis, which is to generate realistic underwater images. During the course of this research, we have explored several alternative sources of datasets but to no avail. Although the VAROS synthetic underwater dataset provides a rich collection of uniform lighting underwater images and their corresponding realistic underwater equivalent, this is not exactly what we need for this work. Ideally, we require real underwater images that are either the same as the in-air uniform lighting images in the case of paired image translation or have similar content with the uniform lighting image in unpaired image translation. This requirement is crucial as we plan to train our model to refine synthetic images to appear like real underwater images by conditioning the model on real underwater images. In the pix2pix [51] scenario, we create the paired training example from the VAROS dataset [29] with underwater images that are generated from a simulation environment. This is apparent in the results generated by the model, as a model cannot learn to introduce real artifacts to refine synthetic images in the absence of real underwater images during training. We select unpaired training samples from the VAROS dataset to train our CoDe method, CycleGAN [9], and SimGAN [8]. This makeshift dataset is intended to test the ability of our method to perform the domain-to-domain translation in the absence of paired training examples, rather than, more broadly speaking generating realistic datasets as both domains are synthetic. The reason behind this makeshift dataset is because of the paucity of real underwater datasets that have similar terrain and little or no marine life or reefs as the VAROS synthetic uniform lighting input. For example, the additional experiment section shows the result of our model when conditioned on real data from the UIEB dataset [30]. We observe that our model is unable to properly segment what style to extract or discard from the real dataset and as a consequence we obtained poor results that transfer the underwater style, reef, and fishes to the synthetic uniform lighting input.

For this work, we plan to investigate unpaired domain-to-domain translation. We see from Figure 5.10 that our model produces remarkable results even in the absence of paired training examples. To solve for the shortcoming of using the VAROS underwater data set to condition our model rather than real datasets, we introduce two new data sets - The UIEB [30] and our custom underwater dataset [53]. In each case, we substitute the folder A underwater dataset above. The results obtained using these real underwater datasets are far from realistic and introduce several unreal artifacts as shown in Figure 5.12. A careful observation of the various artifact reveals several shortcomings. One major limitation is the cornucopia of underwater marine life and colorful coral in the UIEB datasets, forcing our network to introduce this object in a bid to enhance realism. Another downside of the UIEB dataset is the myriad of disparate lighting and coloration in the dataset. This is expected as the dataset is collated from random underwater environments with varying color distributions. Our custom datasets improve slightly on the UIEB dataset in terms of color distribution. We carefully selected images from a video sequence and some underwater environments with similar color distribution. Despite this, our dataset still produces worse results than the VAROS dataset. We surmise that in both cases this failure can be attributed to the real underwater dataset having a mammoth difference in the content as compared to the Uniform synthetic images we aim to translate. For example, we notice that both real underwater datasets have no pipelines similar to what we have in the VAROS synthetic input data, and this poses a problem for the network when learning an ideal mapping between both domains. To tackle this obvious limitation, we need real underwater images that are similar in some sense to the synthetic images from the VAROS platform.

6.2 Model Architecture

The model architecture holds much relevance in the result obtained in the previous chapter. For the paired image-to-image translation, we use two models - autoencoder and pix2pix. The pix2pix produces better results than our naive autoencoder. This is not much of an issue because the core part of this work is on the unpaired image translation method CoDe. For the autoencoder, we use the encoder-decoder architecture with ResNet34 as the encoder and a series of convolutional and transposed convolutional layers as the decoder. The MSE loss is used as an optimization criterion. On the other hand, the pix2pix uses the U-Net autoencoder which has been shown to produce better image generation than the encoder-decoder [55]. Moreso, the pix2pix uses a PatchGAN discriminator to enforce high-level features. These model architectures explain why the pix2pix outperforms our vanilla autoencoder. Our autoencoder, however, has far fewer parameters than pix2pix. Gaty's style transfer doesn't learn a network but rather uses a pre-trained model to extract features from the style and content images and performs gradients on a white noise image. Reproducing the algorithm in SimGAN [8] was a major huddle in this thesis. Using the simplistic generator architecture proposed in the paper produced unrealistic results. The results obtained were more or less an identity mapping of the input. We initially assumed the VAROS dataset is unsuitable for this algorithm, however, applying this to the MPI eye dataset used in the paper didn't produce any better results. After performing a rigorous ablation study including replacing the generation architecture with a more complex architecture, we observe that the generator of our Pytorch implementation after several epochs just learns a one-to-one mapping from input to output. Adjusting the λ value to a very small number to reduce the effect of the L1 loss which functions to preserve the input annotation on the generated output didn't improve the results but rather leads to mode collapse. For the WaterGAN [50] architecture, just im-

plementing the attenuation function produces similar results as shown in the paper, and as a result, we did not bother to delve deep deeper into the proposed network architecture. The CycleGAN [9] and our method have a similar architecture with just a few subtle differences that impact memory and speed. The Architecture of the CycleGAN uses two discriminators and generators to compute the cycle consistent loss and to perform two-sided translation. For our one-sided image translation, we use just a single generator and discriminator and replace the cycle consistent loss with contrastive loss. This makes our approach faster and requires less memory.

6.3 Evaluation Metric

We explore two quantitative metrics to augment the subjective visual evaluation metric. These two metrics are the FID and SSIM. FID calculates the distance between the expected ground truth and the refined generated output from our model. A lower FID indicates that the generated images are more similar to the real images, implying higher image quality. On the other hand, the SSIM ranges from - 1 to 1, with -1 meaning complete dissimilarity and 1 signifying high similarity. Thus, we desire an SSIM score as close to 1 as possible. Comparing the ranking of the generated image quality in Table 5.1 by SSIM and FID, we see the superior judgment of the FID as its ranking is consistent with our visual analysis. A closer look at FID between CycleGAN and our CoDe method underscores the robustness of our approach in outperforming CycleGAN despite having lesser parameters and consequently requiring lesser training time. The 5th column in Table 5.1 shows the FID score on 190 test samples, and our CoDe model with a score of 106 achieves the lowest FID score in comparison with all other baselines. For quantitative analysis, we project both categories of our dataset into a 2D space representation after performing PCA and TSNE dimensionality reduction. According to Figure 5.5, we see that the realistic images in Blue form a cluster around the axis -1.5 and -1.25 while the synthetic input is shifted slightly to the right. On the other hand, the TNSE provides a more distributed cluster, however, in all feature spaces, it is clear that samples from the synthetic input are close to one another (Figure 5.6). The significance of this is that, during the generation of realistic underwater images, we expect our result to occupy a similar space as shown in the plot. The second quantitative approach we adopted was to plot the RGB channels for both the uniform lighting images and the realistic underwater images from the VAROS platform to provide a deeper insight into the color distribution for both categories (Figure 5.4). Notice that both categories of images have disparate distribution. For the uniform lighting example, we see that the RGB histograms have similar peak pixel frequencies, with the peak pixel of the Blue channel occurring around pixel value 150. On the contrary, in the underwater images, the Blue channel has an extremely high pixel count in all 4 samples and these peak frequencies occur between pixel values 0 to 50. What this implies is that by estimating and comparing the pixel of all RGB color bands, we can say categorically which class our generated image belongs to. For instance, if we analyze a generated image and it has a Red channel peak frequency higher than the Blue and Green channels, we can conclude that the generated image lacks realistic underwater properties. This is the case with the SimGAN model.

6.4 Ablation Study

We perform a series of ablation studies to understand the importance of different components in our model as well as enhance the explainability of our model. First, we compared the

result generated using depth and no-depth information in our method. Our observation is that the output generated using depth can model properly underwater attenuation which is consistent with what we expect. Next, we compare the result on the UIEB dataset [30] and our custom data. We observe that both datasets can enforce some minimal underwater color representation but the result is no better than those obtained training on the VAROS dataset. We have discussed the reason for this shortcoming in Section 6.2. However, it is important to reiterate that the model fails to learn the underwater attenuation properly because unlike in VAROS datasets, there is no apparent attenuation in the images from the UIEB and our custom dataset. Also, the model is unaware of what to learn in a complex image structure to improve realism.

6.5 Comparison with Baselines

For the unpaired image translation only the CycleGAN, CUT and our CoDe model produce compelling results that can be categorized as realistic underwater images. Our CoDe model can capture the light attenuation and predict the correct color distribution similar to the ground truth VAROS underwater images as seen in Figure 5.10. The images of the pipelines and seafloor generated by our model correspond to those of the synthetic input image. We acknowledge that the CycleGAN and CUT generated good enough results, however, in some edge cases notice the CycleGAN model struggles to correctly map the input to the realistic underwater domain. For example, in Figure 5.14, we presented a couple of failures of the CycleGAN, CUT and CoDe. In all these cases except for the 3rd row, the CycleGAN erroneously maps rocks from the VAROS dataset to texture and colors indicating pipelines while our model struggles in the first two rows but produced images not far off from the ground truth in the remaining 4 rows. One explanation for this failure is the fact that the unpaired training data used in training the CycleGAN and CoDe do not contain the rough rock samples, rather the biggest rough complex structure is the pipelines. These inadequacies in training data force our model to wrongly learn to generalize any complex structure as a pipeline. Although our results are not perfect, nevertheless, this is an improvement to the results obtained by CycleGAN. We also observed that running some edge cases through the paired pix2pix model produces poor results.

The proof of our CoDe model robustness and generative capability is supported quantitatively by the FID score computed on all 190 test samples. The FID of 106 by our model is the lowest, this is closely followed by CycleGAN then pix2pix and CUT. Although pix2pix is a paired training model, it produces worse results here because this test set contains a collection of easy to difficult edge cases, some of which deviates significantly from the training datasets. A distinctive aspect of our results, when compared to previous baselines, is that our model is able to exploit depth parameters as input and uses half the parameters of CycleGAN to produce a comparable score on the FID benchmark. Our finding indicates that unpaired image translation techniques such as CycleGAN, CUT and CoDe are powerful in generating realistic underwater images as compared to their paired image translation counterpart when the input deviates largely from the training data. We learn that there is a strong correlation between the introduction of depth parameters in the input and improving the realism of underwater images. To be specific, the CUT and CoDe algorithms are similar in that they both use contrastive learning in their loss function and the only major difference is the introduction of depth as an input in CoDe, adjusting the model architecture to account for this 4-channel input and padding the generated image with an extra depth dimension for PatchNCE loss computation during training. This extra depth information significantly

reduced the FID score of the CUT by 26 surpassing CycleGAN and all other baselines.

The creation of realistic synthetic underwater images holds several significant benefits for practitioners in marine robotics. Practitioners often rely on specialized equipment, such as underwater cameras, sensors, and lighting systems. Realistic synthetic images allow practitioners to test and calibrate their equipment in a controlled environment. They can assess the performance and accuracy of their gear, optimize settings, and ensure that the equipment functions properly before deploying it in real-world underwater environments. More so, practitioners in fields such as underwater archaeology, marine biology, and oceanography can benefit from realistic synthetic images for visualization and design purposes. By creating accurate representations of underwater scenes, practitioners can visualize and plan their research projects or expeditions which effectively aid data interpretation, hypothesis formulation, and decision-making processes while mitigating risks and ensuring safety in underwater operations. Currently, generating realistic underwater images is mostly done by developing in-air uniform lighting scenes and then using physics-based image formation models (IFM) to introduce underwater effects and coloration in the environment. This IFM requires domain knowledge of the inherent properties of the underwater environment and long hours of parameter adjustment.

The implication of our finding is that practitioners with no domain knowledge of the underwater environment parameters can now use our CoDe model to introduce realistic underwater effects to uniform in-air images. More so, practitioners can model different underwater environments ranging from sea to ocean by varying the real underwater images used to condition our CoDe model during training. Our findings are equally significant to researchers in marine robotics and technology. Realism in synthetic underwater images allows researchers to assess the performance and generalization capabilities of their algorithms more accurately. If synthetic images closely resemble real-world underwater scenes, it becomes easier to evaluate how well the algorithms will perform in practical applications. Realistic images help researchers identify potential challenges and limitations in their algorithms, leading to improvements and better adaptation to real-world scenarios. Until this time, researchers spend extra long hours trying to model a physics-based model to match their environment of interest. However, with our CoDe model, researchers can save that extra time by exploiting the end-to-end realistic underwater generative capability of our model. Our findings introduce the need for additional qualitative metrics for the underwater environment and underpin the importance of depth information in marine research.

One limitation of our method and all other end-to-end image translation baseline is the requirement for the real underwater images used to condition our model to have similar contents and structure, in some sense, to the uniform input in-air images. This is because, in all unpaired image translation techniques that we explored, the models struggle to properly identify the underwater style when the real images have varying contents that deviate from the input. This is apparent in the results shown in Figure 5.12. Although our model is able to learn underwater coloration, in some instances it introduces some textures that mimic the fish and reefs in the training samples. This limitation forces us to use realistic underwater images, although synthetic, as a makeshift substitute for real underwater images in all the experiments presented in this work. To counter any form of potential bias in this setup, we select sequences from the realistic underwater VAROS folder A that are completely different from the uniform lighting input in folder B in terms of content and structure. In practice, this should not be the case because the entire purpose of this work is to detach from the need of using a physics-based synthetic underwater image. However, for the purpose of this research and to compare our developed CoDe method with other baselines for image translations this is sufficient. Another shortcoming is using a generative adversarial network

for generating realistic underwater images is that although the model learns to effectively transfer the underwater style, it sometimes struggles to preserve the content of the input image. This is apparent in the failed example cases shown in Figure 5.14. The CycleGAN uses the cycle consistent loss to present the input content while our approach uses contrastive loss. In both cases, it is clear that these losses are not sufficient to constrain the generated output because our model produces some results that significantly deviate from the synthetic uniform lighting input in terms of content. For future work, it would be interesting to see the effect of additional terms in the loss function of the model that augments the contrastive loss for preserving the input content in the generated underwater realistic images.

Chapter 7

Conclusion

So far, we have carefully developed our model architecture and explored several experiments, culminating in the results presented in Section 5. These results are intended to answer the core research questions introduced in Chapter 1. This Chapter aims to align these results with the broader context of generating realistic underwater images and provide insights to underpin the significance of our findings to practitioners and researchers in Marine Technology. Furthermore, we acknowledge and discuss the limitations inherent in our method and highlight possible factors that may have influenced these results while pointing out potential directions for future research.

7.1 Conclusion

The core objective of this research work is to develop an end-to-end neural network for generating realistic underwater images. Until this time most of the methods used for the synthesis of underwater images are based on the IFM. Since this method is physics-based, it requires domain knowledge of the inherent underwater parameter and parameter tuning. We examined the effectiveness of using unpaired image translation techniques to transform uniform lighting images into realistic underwater images while conditioning on real underwater images. The experiments designed in this thesis are intended to propel us toward answering the following research questions: **I**. Can we build an end-to-end deep learning model to generate realistic underwater images with better quality or similar to the physics-based realistic underwater images without domain knowledge of the inherent underwater parameters? **II**. Is there a way to effectively preserve the content of the uniform lighting synthetic input while retaining the style of the real underwater images? **III**. Are there any significant improvements in using depth information as the 4th channel of our synthetic uniform lighting input for underwater image generation? **IV**. What objective evaluation metric is suitable for evaluating the quality of our generated realistic underwater images?

To facilitate answering these research questions, we adapted the CUT [10] method to include depth information in the model architecture and training procedure. This model uses contrastive loss and GAN to generate realistic underwater images from uniform lighting input conditioned on real underwater images. The contrastive loss main function is to preserve the content of the synthetic uniform light image. After performing inference on 190 uniform lighting images from the VAROS environment and computing the FID scores, we conclude that our model produced compelling realistic underwater images that can rival the physics-based images from the VAROS synthetic underwater dataset. Our statistical analysis of the FID scores suggests that using contrastive loss to replace the cycle-consistent loss of

CycleGAN resulted in slightly higher FID scores. Although using a depth map to augment the contrastive loss led to a significant drop in the FID score indicating a better performance and generative capability of our model when depth information is provided. Comparing the ranking by subjective visual evaluation of our various models with the FID and SSIM scores in Table 5.1, we deduce that the FID is a robust metric that is consistent with subjective human evaluation and suitable for evaluating the quality of the generated images. In addition, we conditioned our model on our custom real underwater images to investigate the robustness of the model. Our findings imply that our model is able to learn the underwater coloration of the real underwater environment, however, occasionally introduces some texture from fishes and reefs leading to generally poor results. We surmise that our model is unable to attend to the style of interest from the real underwater images. It is possible that outcomes would vary if conditioned on real underwater images with uniform and clear terrain similar to the VAROS environment.

In conclusion, the creation of realistic synthetic underwater images provides numerous advantages for marine robotics practitioners and researchers. These images enable practitioners to test and optimize their specialized equipment in a controlled environment, ensuring accurate performance and functionality before deployment in real-world underwater settings. Additionally, practitioners in fields like underwater archaeology, marine biology, and oceanography can utilize these images for visualization, design, and planning purposes, facilitating data interpretation, hypothesis formulation, and decision-making while ensuring safety and mitigating risks. Currently, the generation of realistic underwater images relies on complex physics-based models and extensive parameter adjustments, requiring domain knowledge of the underwater environment. However, our CoDe model overcomes this limitation by allowing practitioners without underwater domain knowledge to introduce realistic underwater effects to uniform lighting images. The model can be conditioned with various real underwater images, enabling the modeling of different underwater environments. Our findings also have significant implications for researchers in marine robotics and technology, as realistic synthetic images enable more accurate assessment of algorithm performance and generalization capabilities. Realistic images aid in identifying algorithmic challenges and limitations, leading to improvements and better adaptation to real-world scenarios. By leveraging the end-to-end generative capability of our CoDe model, researchers can save time and eliminate the need for laborious physics-based modeling. Furthermore, our findings highlight the importance of additional qualitative metrics for the underwater environment and emphasize the significance of depth information in marine research. While our method exhibits limitations, such as the requirement for similar content in conditioning images and potential content preservation issues, future work can explore additional terms in the loss function to address these challenges and further enhance the generation of realistic underwater images.

7.2 Future Work

Future researchers should consider investigating the impact of the following suggestions. Regardless, our results point to the need for marine researchers to explore using end-to-end neural networks for improving the realism of underwater images.

Content Preservation Enhancement: One area for future investigation is the improvement of content preservation in the generated underwater images. Although our model effectively transfers the underwater style, it sometimes struggles to preserve the content of the input image. To address this, additional terms could be explored and incorporated into

the loss function of the model. These terms could augment the contrastive loss and provide better constraints for preserving the input content in the generated underwater realistic images.

Exploration of Advanced Qualitative Metrics: Our findings emphasize the need for additional qualitative metrics specific to the underwater environment. Future work could focus on developing and refining such metrics to provide a more comprehensive evaluation of the realism and fidelity of synthetic underwater images. These metrics could take into account factors such as water clarity, light diffusion, and object visibility, among others, to ensure a more accurate assessment of the generated images.

Integration with Underwater Robotics and Technology: Building upon our findings, future work can focus on the integration of synthetic underwater images into underwater robotics and technology. By leveraging the realistic images generated by our CoDe model, researchers and practitioners can conduct extensive testing, simulation, and validation of algorithms, systems, and equipment in a virtual environment. This integration can provide valuable insights and contribute to the advancement of underwater robotics, enabling more accurate performance evaluation and enhancing the development of reliable and efficient underwater technologies.

Exploration of Hybrid Approaches: In addition to the end-to-end generative approach presented in our CoDe model, future work can explore hybrid approaches that combine physics-based modeling and deep learning techniques. By integrating the knowledge of underwater physics into the generative process, the resulting synthetic underwater images could exhibit enhanced realism and better adherence to the physical properties of underwater environments. Such hybrid approaches have the potential to further bridge the gap between synthetic and real underwater imagery, improving the accuracy and applicability of generated images in practical underwater applications.

By attending to these suggestions in future research, researchers can advance the field of the generation of realistic underwater images by pushing the boundaries a bit further, empowering practitioners and researchers in marine robotics and related disciplines with more realistic and versatile tools for their work.

Bibliography

[1] S. Impedovo, L. Ottaviano and S. Occhinegro, 'Optical character recognition—a survey,' *International journal of pattern recognition and artificial intelligence*, vol. 5, no. 01n02, pp. 1–24, 1991.

[2] P. Viola and M. Jones, 'Rapid object detection using a boosted cascade of simple features,' in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, Ieee, vol. 1, 2001, pp. I–I.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'Imagenet: A large-scale hierarchical image database,' in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[4] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition,' *arXiv preprint arXiv:1409.1556*, 2014.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, 'Generative adversarial networks,' *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[7] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, 'Hierarchical text-conditional image generation with clip latents,' *arXiv preprint arXiv:2204.06125*, 2022.

[8] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang and R. Webb, 'Learning from simulated and unsupervised images through adversarial training,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2107–2116.

[9] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, 'Unpaired image-to-image translation using cycle-consistent adversarial networks,' in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[10] T. Park, A. A. Efros, R. Zhang and J.-Y. Zhu, 'Contrastive learning for unpaired image-to-image translation,' in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, Springer, 2020, pp. 319–345.

[11] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, 'Image quality assessment: From error visibility to structural similarity,' *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[12] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, 'Gans trained by a two time-scale update rule converge to a local nash equilibrium,' *Advances in neural information processing systems*, vol. 30, 2017.

[13] L. Shen, L. R. Margolies, J. H. Rothstein, E. Fluder, R. McBride and W. Sieh, 'Deep learning to improve breast cancer detection on screening mammography,' *Scientific reports*, vol. 9, no. 1, p. 12 495, 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, 'Attention is all you need,' *Advances in neural information processing systems*, vol. 30, 2017.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML].

[16] T. Karras, S. Laine and T. Aila, *A style-based generator architecture for generative adversarial networks*, 2019. arXiv: 1812.04948 [cs.NE].

[17] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, *High-resolution image synthesis with latent diffusion models*, 2022. arXiv: 2112.10752 [cs.CV].

[18] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, 'Segment anything,' *arXiv preprint arXiv:2304.02643*, 2023.

[19] B. McGlamery, 'A computer model for underwater camera systems,' in *Ocean Optics VI*, SPIE, vol. 208, 1980, pp. 221–231.

[20] J. S. Jaffe, 'Computer modeling and the design of optimal underwater imaging systems,' *IEEE Journal of Oceanic Engineering*, vol. 15, no. 2, pp. 101–111, 1990.

[21] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[22] I. Misra and L. van der Maaten, *Self-supervised learning of pretext-invariant representations*, 2019. arXiv: 1912.01991 [cs.CV].

[23] K. He, H. Fan, Y. Wu, S. Xie and R. Girshick, *Momentum contrast for unsupervised visual representation learning*, 2020. arXiv: 1911.05722 [cs.CV].

[24] T. Chen, S. Kornblith, M. Norouzi and G. Hinton, 'A simple framework for contrastive learning of visual representations,' in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607.

[25] Y. LeCun, Y. Bengio *et al.*, 'Convolutional networks for images, speech, and time series,' *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[26] V.-E. Neagoe, A.-D. Ciotec and G.-S. Cucu, 'Deep convolutional neural networks versus multilayer perceptron for financial prediction,' in *2018 International Conference on Communications (COMM)*, 2018, pp. 201–206. DOI: 10.1109/ICComm.2018.8484751.

[27] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors,' *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[28] M. Mirza and S. Osindero, 'Conditional generative adversarial nets,' *arXiv preprint arXiv:1411.1784*, 2014.

[29] P. G. O. Zwilgmeyer, M. Yip, A. L. Teigen, R. Mester and A. Stahl, 'The varos synthetic underwater data set: Towards realistic multi-sensor underwater data with ground truth,' in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3722–3730.

[30] C. Li, C. Guo, W. Ren, R. Cong, J. Hou, S. Kwong and D. Tao, 'An underwater image enhancement benchmark dataset and beyond,' *IEEE Transactions on Image Processing*, vol. 29, pp. 4376–4389, 2020. DOI: 10.1109/TIP.2019.2955241.

[31] A. Sedlazeck and R. Koch, 'Simulating deep sea underwater images using physical models for light attenuation, scattering, and refraction,' 2011.

[32] H. Blasinski, T. Lian and J. Farrell, 'Underwater image systems simulation,' in *Imaging Systems and Applications*, Optica Publishing Group, 2017, ITh3E–3.

[33] O. Álvarez-Tuñón, A. Jardón and C. Balaguer, 'Generation and processing of simulated underwater images for infrastructure visual inspection with uuvs,' *Sensors*, vol. 19, no. 24, p. 5497, 2019.

[34] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat and T. Rauschenbach, 'Uuv simulator: A gazebo-based package for underwater intervention and multi-robot simulation,' in *OCEANS 2016 MTS/IEEE Monterey*, IEEE, 2016, pp. 1–8.

[35] Y. Song, D. Nakath, M. She, F. Elibol and K. Köser, 'Deep sea robotic imaging simulator,' in *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part II*, Springer, 2021, pp. 375–389.

[36] Y. Hashisho, M. Albadawi, T. Krause and U. F. von Lukas, 'Underwater color restoration using u-net denoising autoencoder,' in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2019, pp. 117–122. DOI: 10.1109/ISPA.2019.8868679.

[37] C. Fabbri, M. J. Islam and J. Sattar, 'Enhancing underwater imagery using generative adversarial networks,' in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7159–7165.

[38] Y. Yu and C. Qin, 'An end-to-end underwater-image-enhancement framework based on fractional integral retinex and unsupervised autoencoder,' *Fractal and Fractional*, vol. 7, no. 1, p. 70, 2023.

[39] G. Kim, S. W. Park and J. Kwon, 'Pixel-wise wasserstein autoencoder for highly generative dehazing,' *IEEE Transactions on Image Processing*, vol. 30, pp. 5452–5462, 2021.

[40] D. P. Kingma and M. Welling, 'Auto-encoding variational bayes,' *arXiv preprint arXiv:1312.6114*, 2013.

[41] W. Xu, S. Keshmiri and G. Wang, 'Adversarially approximated autoencoder for image generation and manipulation,' *IEEE Transactions on Multimedia*, vol. 21, no. 9, pp. 2387–2396, 2019.

[42] N. Koenig and A. Howard, 'Design and use paradigms for gazebo, an open-source multi-robot simulator,' in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, pp. 2149–2154.

[43] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, 'Carla: An open urban driving simulator,' in *Conference on robot learning*, PMLR, 2017, pp. 1–16.

[44] L. A. Gatys, A. S. Ecker and M. Bethge, 'Image style transfer using convolutional neural networks,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.

[45] S. Lee, B. Park and A. Kim, 'Deep learning based object detection via style-transferred underwater sonar images,' *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 152–155, 2019.

[46] J. Johnson, A. Alahi and L. Fei-Fei, 'Perceptual losses for real-time style transfer and super-resolution,' in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, Springer, 2016, pp. 694–711.

[47] X. Zhou, C. Yu, X. Yuan and C. Luo, 'Matching underwater sonar images by the learned descriptor based on style transfer method,' in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2029, 2021, p. 012 118.

[48] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim and J. Choo, 'Stargan: Unified generative adversarial networks for multi-domain image-to-image translation,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.

[49] N. Li, Z. Zheng, S. Zhang, Z. Yu, H. Zheng and B. Zheng, 'The synthesis of unpaired underwater images using a multistyle generative adversarial network,' *IEEE Access*, vol. 6, pp. 54 241–54 257, 2018.

[50] J. Li, K. A. Skinner, R. M. Eustice and M. Johnson-Roberson, 'Watergan: Unsupervised generative network to enable real-time color correction of monocular underwater images,' *IEEE Robotics and Automation letters*, vol. 3, no. 1, pp. 387–394, 2017.

[51] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, 'Image-to-image translation with conditional adversarial networks,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[52] M.-Y. Liu and O. Tuzel, *Coupled generative adversarial networks*, 2016. arXiv: 1606.07536 [cs.CV].

[53] YouTube, Aug. 2010. [Online]. Available: https://www.youtube.com/watch?v=4HgD1Nv36X8&list=LL&index=4.

[54] M. Seitzer, *pytorch-fid: FID Score for PyTorch*, https://github.com/mseitzer/pytorch-fid, Version 0.3.0, Aug. 2020.

[55] O. Ronneberger, P. Fischer and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV].

Generating Realistic Underwater Images Using Contrastive Learning

Kazeem SHAMBA[a], Asgeir SØRENSEN[a], Oscar PIZARRO[a], Ricard MARXER[b], Hacene KARRAD[c], Nicola LUMINARI[c]

[a]Department of Marine Technology - NTNU, Trondheim, Norway

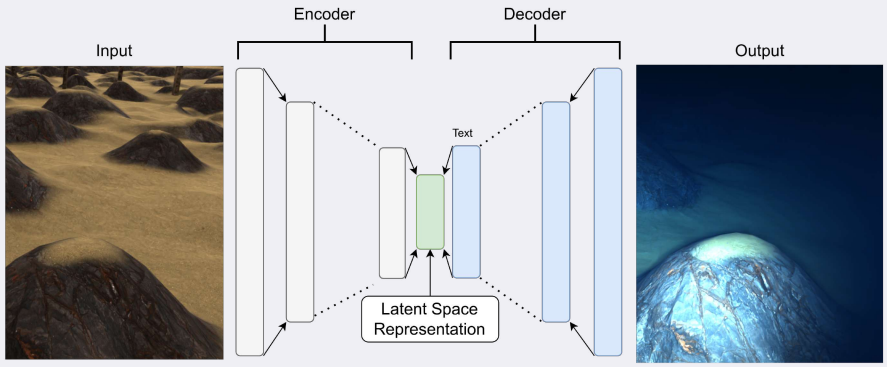
[b]Seatech - Marine Intelligent Robotics, University of Toulon, France [c]Data Science - Alteia, Toulouse, France

Co-funded by the Erasmus+ Programme of the European Union

Introduction

Most of the models we have today are supervised, therefore success or **good generalization** capability of the model, relies heavily on large amounts of labeled training data. Due to the cost, time, and complexity of providing annotations, as the natural world is **unannotated**, there is a need for a learning technique that can **improve the realism** of **synthetic data** from simulated environments. In this thesis, we investigate two major **image-to-image translation** techniques using autoencoders and **contrastive learning** for improving the realism of **synthetic underwater images** and compare our results with several baselines.

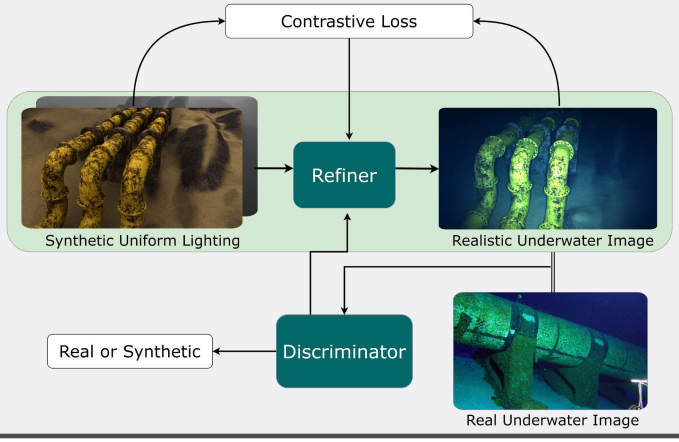
Autoencoder Network - Paired Image Translation



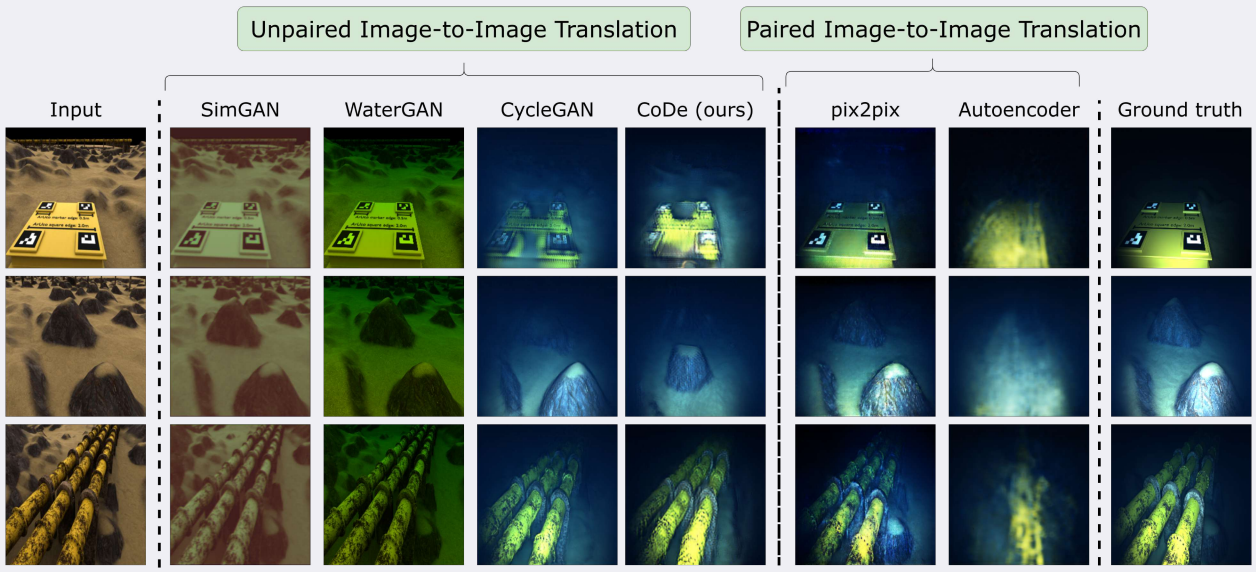
Methods

We explored both **paired** and **unpaired image translation**. First, we build a rather naive image translation model using an **autoencoder** and compare our results with those obtained using the SOTA **pix2pix** [1]. Second, we adapted the CUT [2] framework and adjust the input to account for an extra dimension of **depth information**. We call our method **CoDe (Contrastive+Depth)** for simplicity. The Figure in the next section shows an overview of our CoDe algorithm. We perform a comprehensive comparison with the state-of-the-art in unpaired image-to-image translation as shown in the Results section.

Contrastive Learning - Unpaired Image Translation



Results



Conclusion

In this work, we have explored the benefits of using contrastive learning to improve the realism of synthetic underwater images. We compared the performance with SOTA methods on image translation as shown above. From the above Figure, we derive the following conclusions

- **pix2pix:** This method produced the best results, however, it requires paired data which can be difficult to obtain
- **CycleGAN:** This unpaired image translation technique generated remarkable results
- **CoDe:** our CoDe model produced compelling results comparable with SOTA CycleGAN despite having lesser parameters and consequently lesser training time

References

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2017, pp. 1125–1134.

[2] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," Springer, 2020, pp. 319–345.

Appendix A

Additional Python Codes

A.1 WaterGAN Attenuation Code

```
# water-based attenuation and backscatter
class Attenuation(nn.Module):
    def __init__(self):
        super(Attenuation, self).__init__()
        init_r = torch.randn(1, 1, 1) * 0.01 + 0.35
        self.eta_r = nn.Parameter(init_r)
        init_b = torch.randn(1, 1, 1) * 0.01 + 0.015
        self.eta_b = nn.Parameter(init_b)
        init_g = torch.randn(1, 1, 1) * 0.01 + 0.036
        self.eta_g = nn.Parameter(init_g)

    def forward(self, image, depth):
        eta = torch.cat([self.eta_r, self.eta_g, self.eta_b], dim=0)
        eta_manual = torch.tensor([[[0.00008]],[[0.000015]],[[0.01]]])
        print(eta.shape)
        print(eta_manual.shape)
        eta_d = torch.exp(-depth*eta_manual)
        h0 = image * eta_d
        return h0

attenuation = Attenuation()
```

A.2 Underwater Colour Distribution Code

```
#Loading the crab images

impath = pathlib.Path("./Varos_data/air/seq01_veh0_camM0_B-00000260.png")
im_gray = utils.read_image_gray(impath)
im = utils.read_image(impath)

gray_crab = im[:,:,2]
```

```
hist, hist_centers = histogram(im[:,:,0])
hist1, hist_centers1 = histogram(im[:,:,1])
hist2, hist_centers2 = histogram(im[:,:,2])

fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].imshow(im)
axes[0].axis('off')
axes[1].plot(hist_centers, hist, 'b', lw=2)
axes[1].plot(hist_centers1, hist1, 'g', lw=2)
axes[1].plot(hist_centers2, hist2, 'r', lw=2)

axes[1].set_ylabel('Frequency')
axes[1].set_xlabel('Intensity')
plt.xlim([0, 250])
plt.ylim([0, 40000])
plt.show()

impath = pathlib.Path("./Varos_data/water/seq01_veh0_camM0_A-00000260.png")
im_gray = utils.read_image_gray(impath)
im = utils.read_image(impath)

gray_crab = im[:,:,2]
hist, hist_centers = histogram(im[:,:,0])
hist1, hist_centers1 = histogram(im[:,:,1])
hist2, hist_centers2 = histogram(im[:,:,2])

fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].imshow(im)
axes[0].axis('off')
axes[1].plot(hist_centers, hist, 'b', lw=2)
axes[1].plot(hist_centers1, hist1, 'g', lw=2)
axes[1].plot(hist_centers2, hist2, 'r', lw=2)

axes[1].set_ylabel('Frequency')
axes[1].set_xlabel('Intensity')
plt.xlim([0, 250])
plt.ylim([0, 40000])
```

A.3 Feature Space of Both Classes Code

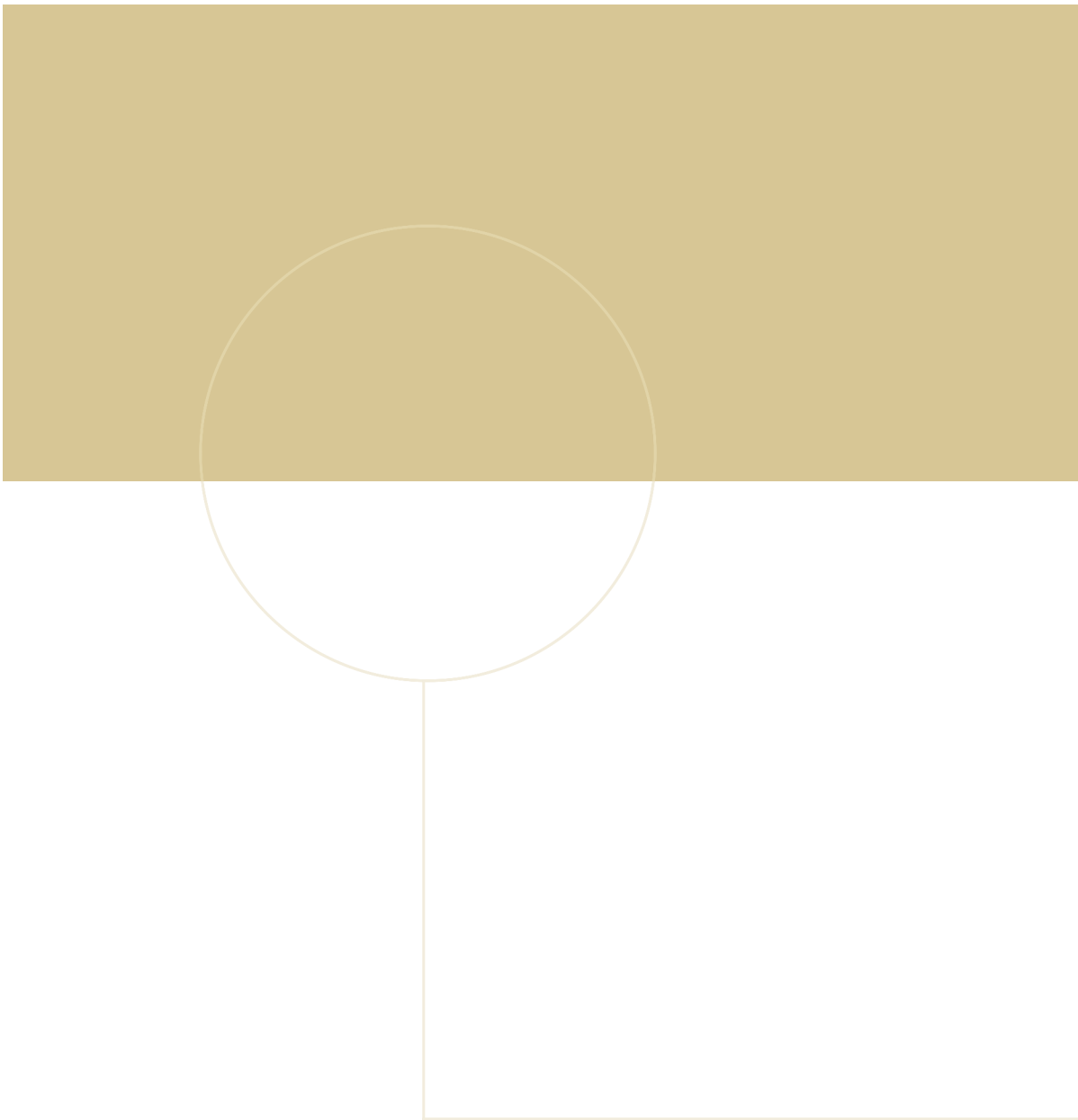
```
codes = codes.cpu().detach().numpy()
codes_pca = PCA(n_components=2).fit_transform(codes)
codes_tsne = TSNE(n_components=2, learning_rate='auto', init='random').fit_transform(codes)

codesr = model_autoresnet.encoder(real_images[0])
codesr = codesr.cpu().detach().numpy()
codes_pcar = PCA(n_components=2).fit_transform(codesr)
codes_tsner = TSNE(n_components=2, learning_rate='auto', init='random').fit_transform(codesr)
```

```
plt.figure(figsize=(9, 5))
plt.scatter(codes_pca[:,0], codes_pca[:,1], color="darkorange", lw=2,label="Synthetic Input",
plt.scatter(codes_pcar[:,0], codes_pcar[:,1], lw=2, label="Realistic Underwater", s=2)

# plt.scatter(codes_tsne[:,0], codes_tsne[:,1], color="darkorange", lw=2,label="Synthetic Inpu
# plt.scatter(codes_tsner[:,0], codes_tsner[:,1], lw=2, label="Realistic Underwater", s=2)

plt.legend(loc=1)
plt.show()
```



NTNU

Norwegian University of
Science and Technology