Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Application of Neural Radiance Fields in visual based subsea maintenance

by

Paul Maximilian Setinek

Email: paul.setinek22@imperial.ac.uk
GitHub username: acse-pms122
Repository: https://github.com/ese-msc-2022/irp-pms122

Supervisors:

Dr. Lukas Mosser (Aker BP)
Dr. Lluis Guasch (Imperial College London)

September 2023

Acknowledgements

I would like to express my deepest appreciation to Dr. Lukas Mosser, whose invaluable assistance throughout this project cannot be overstated. His experience, enthusiasm and thought-provoking discussions elevated the level of this research. I would also like to extend my deepest gratitude to Dr. Lluis Guasch for his valuable advice that guided me throughout the project. I am also grateful to Dr. Edmary Altamiranda Maldonado and Jarle Marius Solland, MSc. They introduced me to the area of subsea maintenance and were pivotal in aligning this project with Aker BP. Additionally, I extend my gratitude to the maintainers of the nerfstudio project. Their assistance and frequent discussions have been invaluable in deepening my understanding of this complex field. Last but not least, I want to express my heartfelt thanks to my family, who have been a constant source of support throughout my entire master's degree.

Abstract

In the rapidly evolving field of Neural Radiance Fields (NeRFs), numerous approaches are increasingly excelling at the task of novel view synthesis. This project explores the application of NeRFs within the subsea maintenance domain, an area that presents both technical and economic challenges. The primary contribution of this work is a modular implementation of a subsea-specific NeRF approach suited for the subsea maintenance domain and built-upon a well-maintained open-source framework. Empirical results indicate that the implemented model is significantly faster to train and infer while performing at a comparable, if not superior, quality level compared to the current state-of-the-art (SOTA) NeRF model. Despite the increasing amount of research in NeRFs, current literature lacks a comprehensive analysis of the intricate aspects of subsea-specific approaches. Accompanied by extensive experimentation, this work addresses this gap by illuminating several previously ambiguous or unrevealed aspects of NeRFs in the subsea domain. It paves the way for future researchers to build on this foundation, further enhancing the efficacy and understanding of these models.

Contents

List of Abbreviations

IMR Inspection, Maintenance & Repair

IPE Integrated Positional Encoding

IRP Independent Research Project

LLFF Local Light Field Fusion

LPIPS learned perceptual image patch similarity

MLP Multi Layer Perceptron

MSE Mean Squared Error

MVS Mulit-View Stereo

NDC Normalized Device Coordinates

NeRF Neural Radiance Field

PDF Probability Density Function

PE Positional Encoding

PSNR peak-signal-to-noise-ratio

ROV Remotely Operated Vehicle

SfM Structure from Motion

SHE Spherical Harmonics Encoding

SOTA state-of-the-art

SSIM structural similarity index measure

W&B Weights & Biases

1 Introduction

Inspection, Maintenance & Repair (IMR) units are increasingly relying on visual data collected from Remotely Operated Vehicles (ROVs) in order to diagnose subsea infrastructure [1]. Currently, this area is dominated by classical computer vision workflows featuring photogrammetry methods like Structure from Motion (SfM) and Mulit-View Stereo (MVS). However, the emerging field or NeRFs offers exciting opportunities within this application area.

One of the primary benefits of NeRFs is their exceptional performance in image reconstruction and novel view synthesis [2, 3, 4]. In the context of IMR, this characteristic can be leveraged to inspect underwater infrastructure from angles that might not have originally been captured by the ROVs. This added flexibility can be crucial, given that some areas might be inaccessible or not considered relevant at the time of data collection. Certain subsea NeRF approaches can distinguish between objects and regions that contain only water within a scene [5]. This functionality is also beneficial to subsea maintenance, as it allows for filtering of water induced effects like backscatter and attenuation without the need for pre- or post-processing of data. Additionally, NeRF models offer the capability to extract geometry from the captured scenes, thus presenting a possible alternative to SfM and MVS. [1]

Furthermore, the field of NeRFs is still relatively young and experiencing rapid development that can be leveraged in the future. Since the first model was published in 2020 [6], more than 450 preprints and publications have been released with more than 210 source codes on GitHub. (as of August 2023)[2]

The idea behind NeRFs is to represent a 3D scene as a continuous function approximated by a neural network, specifically a Multi Layer Perceptron (MLP). This approach offers an advantage in memory consumption compared to methods that store the appearance of a 3D scene explicitly.[3] The first NeRF publication suggests that the method yields a relative compression factor of $3000\times$ [6, p. 13] in storage size, compared to then SOTA methods that use voxel grids such as Local Light Field Fusion (LLFF) [8]. However, one has to emphasize the time versus space trade-off at this point. It can require long training times (up to the order of days, depending on the method and hardware) to train a NeRF compared to voxel-based methods that can process a small input dataset within minutes.

That said, there has been a lot of progress in the field aimed at making NeRF training and inference more efficient. Instant-ngp [3], for example, introduces innovative techniques that efficiently encode input data. This allowed the authors to shrink the MLP down significantly, making it faster to train and query while performing on par with current SOTA methods.

The NeRF publications focusing on the subsea domain are limited. Only two specific publications, dubbed SeaThru-NeRF [5] and Water-NeRF [9], propose approaches tailored to this application area. These two approaches focus on subsea image reconstruction and novel view synthesis. Despite their significant contributions, they come with inherent limitations. Notably, these models struggle with scenes that are captured from 360-degree views. Furthermore, they require long training times. These shortcomings are critical and must be addressed when

[1] It should be noted, however, that the utility of SfM and MVS cannot completely be substituted by NeRFs. This is because the initial camera pose estimation, a preprocessing step essential for training NeRF models, is typically performed using SfM and MVS techniques.

[2] Statistics taken from `https://paperswithcode.com/method/nerf`.

[3] Technically, the volumetric scene representation of NeRFs is also explicit. However, in literature NeRF models are often referred to as implicit models. The authors of [7] put it in the following well-describing words: *"However, NeRF-like volumetric representations are not necessarily implicit — because the output of the network is density and colour, the geometry of the scene is parameterized by the network explicitly, not implicitly. Despite this, it is common in the literature for these models to still be called "implicit", perhaps in reference to the fact that the geometry of the scene is defined "implicitly" by the weights of a neural network."*

incorporating NeRFs into IMR workflows.

At the time of project initiation, no public implementations of NeRFs for subsea scenes were available. Therefore, it was agreed to focus on creating an implementation of the SeaThru-NeRF approach. During development, I made certain justified modifications and augmentations to the original publication to address the above mentioned limitations and align the implementation to the specific requirements of the subsea maintenance domain.

This report starts by revising the theory behind NeRFs in general and continues to make the transition towards models in the subsea domain. It outlines the novelties of my implementation and provides a detailed discussion of experimental results on various scenes. It sheds light on strengths and weaknesses of the model, such as parameter sensitivity and generalisation. Additionally, it outlines opportunities and limitations concerning the subsea maintenance domain.

2 NeRF Theory

I start with a comprehensive summary of a conventional NeRF pipeline, subsequently diving into the intricacies of the volumetric rendering technique. Then, I present adaptations and innovations specific to the subsea domain.

2.1 Fundamentals

The fundamental principle underlying NeRFs is to represent a scene as a continuous function that maps a position, $\mathbf{x} \in \mathbb{R}^3$, and a viewing direction, $\boldsymbol{\theta} \in \mathbb{R}^2$, to a colour $\mathbf{c} \in \mathbb{R}^3$ and volume density $\sigma \in \mathbb{R}^1$.[4] As neural networks can serve as universal function approximators, the authors of [6] suggest to approximate this continuous scene representation with a simple MLP $F_\Theta : (\mathbf{x}, \boldsymbol{\theta}) \rightarrow (\mathbf{c}, \sigma)$.

A conventional NeRF pipeline can be summarized as the following approach:

Sampling of points: Shoot camera rays through the scene to generate a set of 3D points. Literature presents various approaches in order to sample points that are relevant to the final image. Most SOTA methods use proposal networks as explained in Section 2.3.

Network pass: Use the sampled points and corresponding viewing directions as inputs to F_Θ to obtain corresponding density and colour values.

Volumetric rendering: Use the volumetric rendering technique described in Section 2.2 to accumulate the network outputs into a pixel colour value.

Backpropagation: As all the operations above are differentiable, gradient descent can be used to optimize the model weights. The aim is to minimise the Mean Squared Error (MSE) of the pixel colour values.

2.2 Volumetric Rendering

The principles behind the process of volumetric rendering are well established in classical computer graphics pipelines [10]. The authors of [6] and most other NeRF based approaches render the expected colour $\mathbf{C}(\mathbf{r})$ of a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o} \in \mathbb{R}^3$ is the origin of the camera, $\boldsymbol{d} \in \mathbb{R}^3$ is the viewing direction as a 3D unit vector and $t \in \mathbb{R}_+$ is the distance along the ray, as follows:

[4] In most implementations and the derivation provided in this report the viewing direction is actually expressed as a 3D Cartesian unit vector $\boldsymbol{d}$.

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,. \tag{1}$$

In the equation above, t_n and t_f stand for the near and far bounds of the ray, respectively. $T(t)$ denotes the accumulated transmittance along ray $\mathbf{r}(t)$ from t_n to t. It can be written as a continuous function as follows:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \tag{2}$$

In order to compute the integrals above, quadrature can be used as a numerical estimation. This approximation is discussed in [11] and reads:

$$\hat{\boldsymbol{C}}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right). \tag{3}$$

Figure 1 provides a visualisation of a discretized camera ray.

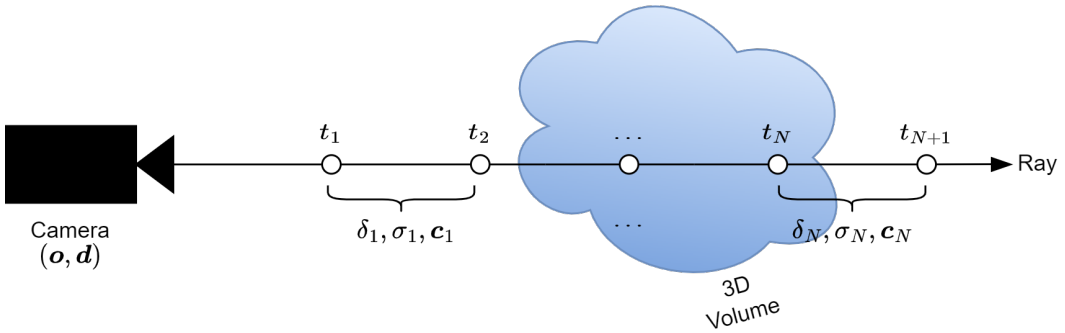


Figure 1: A schematic visualisation of a discretized ray. The continuous ray $\boldsymbol{r}$ gets split up into a set of N intervals $I_i = [t_i, t_{i+1}]$. Density σ_i and colour $\boldsymbol{c}_i$ are assumed to be constant along each interval. Those values are obtained by querying network F_Θ at the midpoint of interval I_i. In Equation (3), δ_i stands for the length of interval I_i and can be calculated via $\delta_i = t_{i+1} - t_i$.

As Equation (3) is fully differentiable, the weights Θ of the underlying neural network can then be trained using a reconstruction loss on the rendered pixels. Many NeRF approaches use a pixel-wise error that can be written as follows:

$$\mathcal{L}_{\text{recon}}(\hat{\boldsymbol{C}}, \boldsymbol{C}^*) = \left\|\hat{\boldsymbol{C}} - \boldsymbol{C}^*\right\|^2, \tag{4}$$

where $\hat{\boldsymbol{C}}$ is the colour rendered with Equation (3) and $\boldsymbol{C}^*$ is the ground truth pixel colour.[5]

[5]The losses in this work are written per ray/pixel. During training, they are averaged over the batch as I am using mini-batch gradient descent to optimize the model parameters.

2.3 Model Derivation for the Subsea Domain

This research project focuses on the application of NeRFs within the subsea domain, where particles travel through water and not air. Hence, an approach with a modified image formation model and network architecture needs to be used in order to account for the effects of scattering media. This section provides an overview of the image formation model and network architecture proposed in SeaThru-NeRF [5]. This derivation forms the basis for the implementation developed over the course of this research project.

The authors of [5] combine the fundamentals of NeRFs with the following underwater image formation model proposed in [12]:

$$I = \overbrace{\underbrace{J}_{\text{colour}} \cdot \underbrace{\left(e^{-\beta^D(\mathbf{v}_D)\cdot z}\right)}_{\text{attenuation}}}^{\textit{direct}} + \overbrace{\underbrace{B^\infty}_{\text{colour}} \cdot \underbrace{\left(1 - e^{-\beta^B(\mathbf{v}_B)\cdot z}\right)}_{\text{attenuation}}}^{\textit{backscatter}}, \tag{5}$$

where I represents the image captured by the camera with range z, J is the clear image without any water effects such as attenuation and backscatter, and B^∞ represents the backscatter water colour at depth infinity. The two components that characterize the effects of the medium are the attenuation coefficient $\beta^D(\mathbf{v}_D)$ and the backscatter coefficient $\beta^B(\mathbf{v}_B)$. The vectors $\mathbf{v}_D$ and $\mathbf{v}_B$ stand for the dependencies of those coefficients on range, object reflectance, spectrum of ambient light, the camera's spectral response, and the physical scattering and beam attenuation coefficients of the water, all of which are wavelength-dependent [5].

As NeRFs need a discrete and differentiable volumetric rendering equation, the authors of [5] propose the following formulation:[6]

$$\hat{\boldsymbol{C}}(\mathbf{r}) = \sum_{i=1}^{N} \hat{\boldsymbol{C}}_i^{\text{obj}}(\mathbf{r}) + \sum_{i=1}^{N} \hat{\boldsymbol{C}}_i^{\text{med}}(\mathbf{r})\,. \tag{6}$$

Compared to Equation (3), this formulation features an object and a medium part contributing towards the final rendered pixel colour $\hat{\boldsymbol{C}}(\boldsymbol{r})$. Those two components are given by:[7]

$$\hat{\boldsymbol{C}}_i^{\text{obj}}(\mathbf{r}) = T_i^{\text{obj}} \cdot \exp(-\boldsymbol{\sigma}^{\text{attn}} t_i) \cdot \left(1 - \exp(-\sigma_i^{\text{obj}} \delta_i)\right) \cdot \mathbf{c}_i^{\text{obj}} \tag{7}$$

$$\hat{\boldsymbol{C}}_i^{\text{med}}(\mathbf{r}) = T_i^{\text{obj}} \cdot \exp(-\boldsymbol{\sigma}^{\text{bs}} t_i) \cdot \left(1 - \exp(-\boldsymbol{\sigma}^{\text{bs}} \delta_i)\right) \cdot \mathbf{c}^{\text{med}}\,, \tag{8}$$

$$\text{where } T_i^{\text{obj}} = \exp\left(-\sum_{j=0}^{i-1} \sigma_j^{\text{obj}} \delta_j\right). \tag{9}$$

The above equations contain five parameters that are used to describe the underlying scene: object density $\sigma_i^{\text{obj}} \in \mathbb{R}^1$, object colour $\mathbf{c}_i^{obj} \in \mathbb{R}^3$, backscatter density $\boldsymbol{\sigma}^{\text{bs}} \in \mathbb{R}^3$, attenuation density $\boldsymbol{\sigma}^{\text{attn}} \in \mathbb{R}^3$ and medium colour $\mathbf{c}^{\text{med}} \in \mathbb{R}^3$.

As this scene parameterization includes more parameters than the classical NeRF approach presented in Sections 2.1 and 2.2, the authors of [5] propose a novel model architecture that is split up into a proposal, an object and a medium network. The idea of proposal networks was first introduced in [2]. Their purpose is to sample points in regions of the scene that contribute

[6] For a detailed derivation of this discretized rendering equation, see Sections 4.1.-4.3. of [5].

[7] In Equations (7) and (8), t_i stands for the distance from the camera to the starting point of Interval I_i, as visualised in Figure 1.

most to the final image. It has a similar architecture to the object network but only predicts object densities, σ_i^{obj}. Additionally, it contains a Probability Density Function (PDF)-Sampler that samples positions from the distribution of the sequence of object weights $\mathbf{w} = \{w_i^{\text{obj}}\}_{i=1}^{N}$.[8] A visualisation of the proposed architecture is provided in Figure 2.

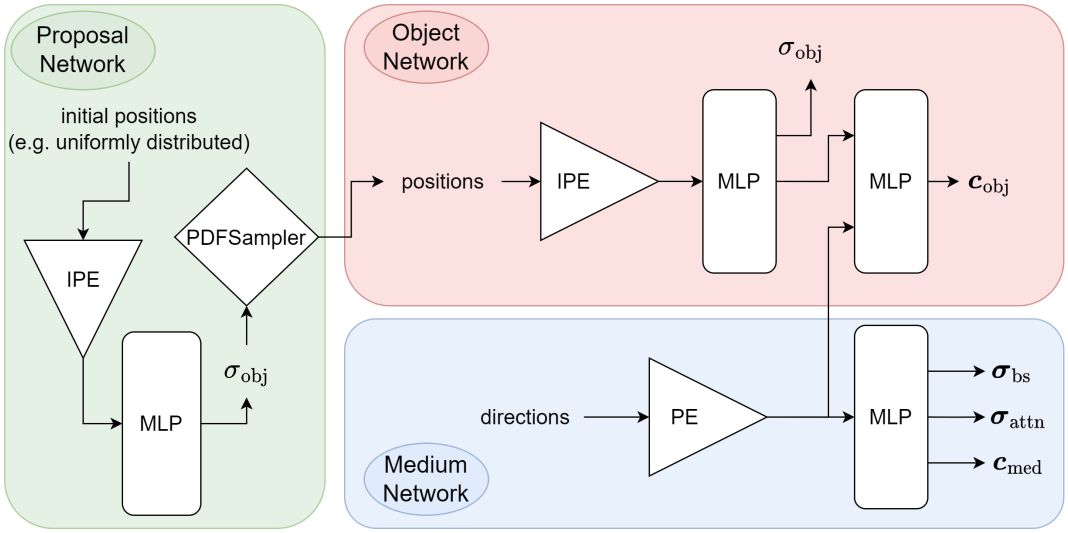


Figure 2: Model architecture of SeaThru-NeRF [5]. The proposal network samples relevant positions. The object network computes σ^{obj} using those positions encoded via Integrated Positional Encoding (IPE), a technique introduced in [13]. To compute $\mathbf{c}^{obj}$ it also uses the viewing directions encoded via Positional Encoding (PE), first introduced in [6]. The medium network only uses the encoded viewing directions in order to compute $\boldsymbol{\sigma}^{\text{bs}}$, $\boldsymbol{\sigma}^{\text{attn}}$ and $\mathbf{c}^{\text{med}}$. This architecture constrains the medium properties to be constant for all t along a single ray $\boldsymbol{r}(t)$.

In order to train the model weights of this architecture, the authors propose a loss function that is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{recon}}(\hat{\boldsymbol{C}}, \boldsymbol{C}^*) + \mathcal{L}_{\text{prop}}(\mathbf{w}_{\text{prop}}, \mathbf{w}) + \lambda \mathcal{L}_{\text{acc}}(T^{\text{obj}}) \,, \tag{10}$$

where $\hat{\boldsymbol{C}}$ is the rendered colour using Equation (6), $\boldsymbol{C}^*$ is the ground truth pixel colour, $\mathbf{w}$ is the sequence of object weights obtained from the object network and $\mathbf{w}_{\text{prop}}$ is the sequence of object weights obtained from the proposal network.

The first component, $\mathcal{L}_{\text{recon}}(\hat{\boldsymbol{C}}, \boldsymbol{C}^*)$, is a weighted MSE loss originally introduced in [14], which penalizes wrong predictions in dark areas of the image more heavily. It is defined as:

$$\mathcal{L}_{\text{recon}}(\hat{\boldsymbol{C}}, \boldsymbol{C}^*) = \left(\frac{\hat{\boldsymbol{C}} - \boldsymbol{C}^*}{\text{sg}(\hat{\boldsymbol{C}}) + \epsilon} \right)^2 , \tag{11}$$

where $\text{sg}(\cdot)$ stands for stop-gradient. This means that the scaling factor gets detached from the computational graph so it does not get optimized during backpropagation.

The second loss component, $\mathcal{L}_{\text{prop}}(\mathbf{w}_{\text{prop}}, \mathbf{w})$, penalizes the mismatch between the distributions of object weights from the proposal network and the object network. A detailed derivation of this loss component can be found in [2].

[8] Object weights are calculated via $w_i^{\text{obj}} = T_i^{\text{obj}} \cdot \left(1 - \exp(-\sigma_i^{\text{obj}} \delta_i)\right)$.

The third loss component, $\mathcal{L}_{\text{acc}}(T^{\text{obj}})$, enforces binary separation between points in the scene that contain an object and those that contain only medium. This is necessary for the model to converge to a physically correct representation of the scene that eventually allows it to filter out water effects from the scene if wanted.[9] To achieve this, the authors of [5] adopt the loss of [15] and propose a prior on the object transmittance of each point along a ray to be either zero or one. This prior can be formulated as follows:

$$\mathbb{P}(x) = e^{-\frac{|x|}{0.1}} + \beta e^{-\frac{|1-x|}{0.1}} \,. \tag{12}$$

with the negative log-likelihood being the loss:[10]

$$\mathcal{L}_{\text{acc}}(T^{\text{obj}}) = -\log \mathbb{P}(T^{\text{obj}}) \,. \tag{13}$$

The contribution of this loss can be controlled by the hyperparameter λ, whereas the shape of the loss function can be controlled by the hyperparameter β. Those two hyperparameters are important to ensure a correct separation of medium and object and a detailed discussion of β is provided in Section 3.4.

3 Contributions

A significant challenge throughout the project was given due to the lack of an available reference implementation. The original Seathru-NeRF publication [5] furthermore references numerous other publications which the authors used as reference without being explicit about their own implementation and hyperparameters. Throughout the development, I implemented several critical modifications diverging from the publication's description in order to ensure the model's convergence. Furthermore, I augmented the approach with features specific to the requirements of the subsea maintenance domain. This section outlines the disparities between my implementation, the SeaThru-NeRF publication [5], and the authors' official GitHub codebase, which was eventually released in mid-July. [11]

3.1 Scene Contraction

Visual data for subsea maintenance is typically acquired from various angles, often covering a full 360-degree range. The original SeaThru-NeRF only works for forward-facing scenes [5]. I however extend the approach to work for scenes that are unbounded in all directions, in order to adapt it to the requirements of the subsea maintenance domain.

The authors of [5] use Normalized Device Coordinates (NDC) which converts the camera axis (mostly referred to as the z-axis) to be linear in disparity (inverse distance). This is a way to warp a scene that is infinitely deep in one direction into a bounded cube [16].

[9] A detailed discussion of a case where this separation fails is given in Section 5.1.

[10] This loss is computed for each sample. It is then averaged over all samples along all rays of the batch during training.

[11] The official source code can be found at: `https://github.com/deborahLevy130/seathru_NeRF`.

In contrast, I use the scene contraction introduced in [2], which is defined as follows:

$$f(x) = \begin{cases} x, & \text{if } \|x\| \leq 1 \\ (2 - \frac{1}{\|x\|}) \cdot \frac{x}{\|x\|}, & \text{if } \|x\| > 1 \end{cases} \tag{14}$$

Depending on the norm used in the above equation, different spatial distortions can be realised. I choose to use the $\boldsymbol{L}_\infty$ norm, as it is suggested to be used when working with grid-based encoders as in [3]. Using this norm, the camera rays and therefore the scene essentially get warped inside a cube of side length four, which makes it possible to process scenes that are unbounded in all directions.

3.2 Encoders

In commercial environments, computational resources are at a premium and fast results are often imperative. Therefore, I adopt input encodings that substantially increase the training and inference speed of my approach, whilst maintaining visual quality.

In general, encoding the input vectors before passing them through the MLPs has shown to be crucial in order to capture high-frequency variations in colour and geometry [6]. This finding is consistent with [17] where the authors outline that deep neural networks have an inclination to learn functions with lower frequency content. Additionally, they demonstrate that transforming the inputs into a higher-dimensional space with the use of high-frequency functions before they are processed by the network can lead to a more accurate fit for data characterised by high-frequency variations. Consequently, leveraging input encoding has become a widely accepted practice in the NeRF domain.

As shown in Figure 2, the authors of [5] use Integrated Positional Encoding (IPE) to encode positions and Positional Encoding (PE) to encode viewing directions. In contrast to that, I adopt trainable Multiresolutional Hash Encodings to encode positions, a technique first introduced in [3]. This type of encoding allows for slimmer MLPs, leading to a substantial reduction in training and inference times whilst maintaining visual quality. This efficiency increase is quantified in Section 5.3. To encode viewing directions, I incorporate Spherical Harmonics Encoding (SHE), an advancement introduced in [18].

3.3 Network Architecture

Using encoded viewing directions as inputs to the MLP that predicts colour is common in NeRF architectures. This is also adopted in the original SeaThru-NeRF publication [5]. The authors of [6] suggest that this is required to guarantee multiview consistency as the same object can have different colours when looking at it from different angles (e.g. shadows).

However, through various experiments during the development of this implementation, I found this not to be necessary. This could be explained by the fact that attenuation plays a part in the rendered object colour (see Equation (7)) and considering this, there is still an (indirect) influence of the viewing direction on object colour for my approach. Figure 3 shows the architecture of my implementation.[12]

[12]Note that while the authors of [5] propose to use the viewing direction as an input to the object MLP in their publication, their code diverges from this description.

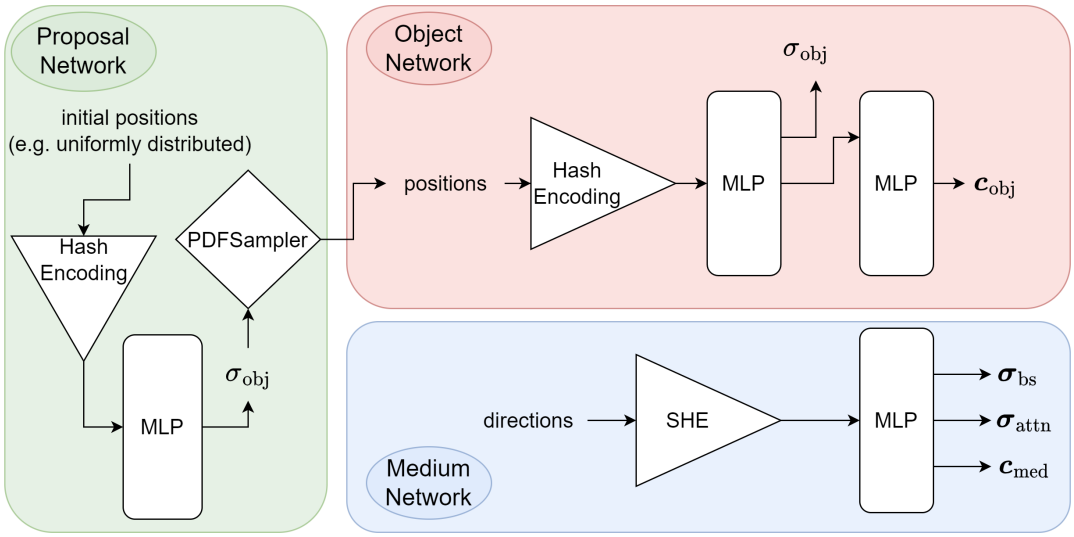


Figure 3: Implemented model architecture.[13]Note the different encoding techniques and the missing link from the viewing direction to the second MLP of the object network compared to Figure 2.

3.4 Separation of Object and Medium

Throughout the developmental experiments for this implementation, it became evident that the right choice of hyperparameter β from Equation (12) is crucial to reach a physically correct separation between object and medium components within a scene. While the authors of [5] do not mention this hyperparameter in their publication, they use $\beta = 6$ in their codebase. To grasp the concept of this hyperparameter's influence on the loss function, Figure 4 shows different loss functions, depending on β.

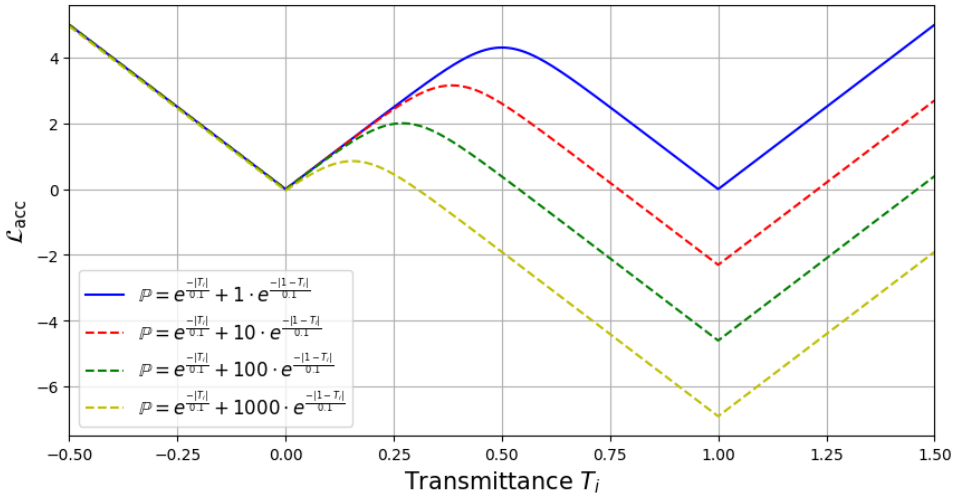


Figure 4: Influence of hyperparameter β on shape of loss function (13). Using $\beta > 0$ leads to a nice side effect concerning the interpretability of the model: When the mean of $\mathcal{L}_{\text{acc}}$ over the batch is negative, it is an indication that the model perceives some areas of the scene to only contain water. This behaviour is desired with subsea scenes.

Detailed experimental results concerning this hyperparameter are discussed in Section 5.1.

[13]To be exact, I actually employ two proposal networks, as it leads to better results. These two networks are sequentially arranged and are jointly optimized based on the sum of their respective loss functions, $\mathcal{L}_{\text{prop}}(\mathbf{w}_{\text{prop}}, \mathbf{w})$. A detailed discussion on the number of proposal samplers is given in Section 5.2.

4 Implementation

For the implementation of my approach, I built upon nerfstudio.[14] I chose this open-source library because it offers modularized components allowing researchers to easily develop, test and interpret new NeRF approaches.

Within nerfstudio, tiny-cuda-nn[15] is adopted for a fast implementation of MLPs. As a consequence, this research needed to be conducted on a CUDA compatible device. As my implemented model uses approximately 23GB of memory, I opted to use the departmental DGX A100 platform, equipped with Nvidia A100 GPUs with 40GB of RAM. A detailed guide on setting up an appropriate environment on this system is provided in the README of my GitHub repository.

To track my experiments and inspect the training process, I used Weights & Biases (W&B) as it is nicely integrated into nerfstudio offering easy customization and extension. An example of some training graphs and evaluation images is shown in Appendix A.

Another tool I heavily used that proved invaluable throughout the research process is the remote development extension within VS Code.[16] It greatly streamlined development, offering the familiar graphical interface on the remote server. This tool allowed me to edit large chunks of source code, which can become very tedious in editors like nano or vim, without having to develop locally and having to push and pull for every change. Furthermore, the extension facilitated the display of renders, making it redundant to transfer them to my local machine each time via SCP.

5 Results

Given the novelty of the subsea NeRF domain, suitable datasets for experiments are notably scarce. For the evaluation of this method, I use the subsea dataset provided by the authors of SeaThru-NeRF [5]. Additionally, Section 6 focuses on the applicability of this method to the subsea visual maintenance domain and explores additional synthetic datasets more suitable to the application area.

Throughout the model's development, experiments were systematically incorporated into the process. This iterative approach ensured that, by the end of the developmental phase, the most critical hyperparameters were evident. Given the challenges associated with traditional hyperparameter optimization methods for time-intensive deep learning models such as the underlying one, it was necessary to constrain the search space for the final implementation's hyperparameter sensitivity analysis. As a result, I centred my sensitivity evaluation on the parameters that appeared most influential during the development phase. Table B.1 offers a comprehensive summary of the executed experiments and their respective outcomes.

The following subsections focus on the most important insights of the conducted experiments in depth. Additionally, an evaluation of my implementation compared to the official SeaThru-NeRF implementation [5] is given.

5.1 Prior Formulation

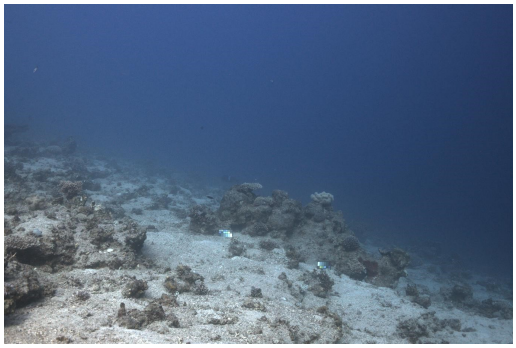
First and foremost, the experimental results indicate that the choice of hyperparameter β from Equation (12) is crucial to ensure a correct separation between object and medium components

[14] https://github.com/nerfstudio-project/nerfstudio/

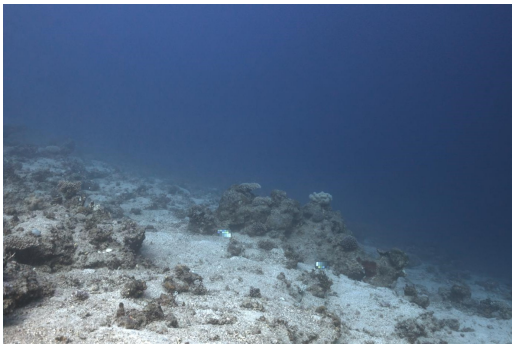
[15] https://github.com/NVlabs/tiny-cuda-nn

[16] https://code.visualstudio.com/docs/remote/remote-overview

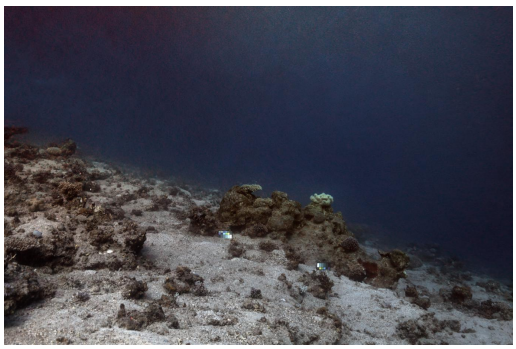
in the scene. Figure 5 shows the reconstruction results of an evaluation image after training the model with $\beta = 10$.



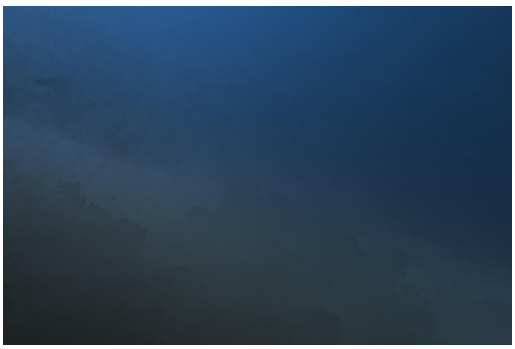
(a) Ground truth image.



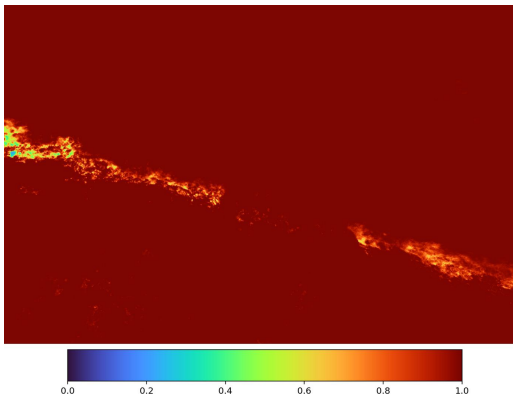
(b) Reconstructed image.



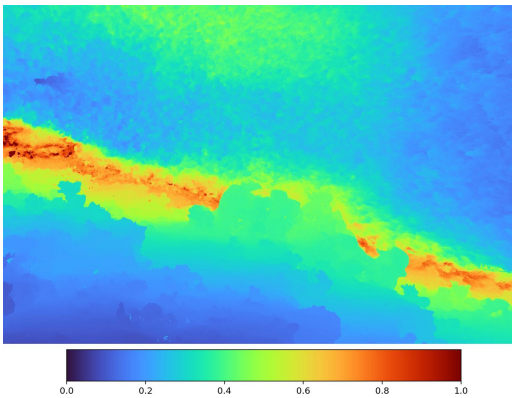
(c) Clear image.



(d) Backscatter.



(e) Object weight accumulation.



(f) Depth.

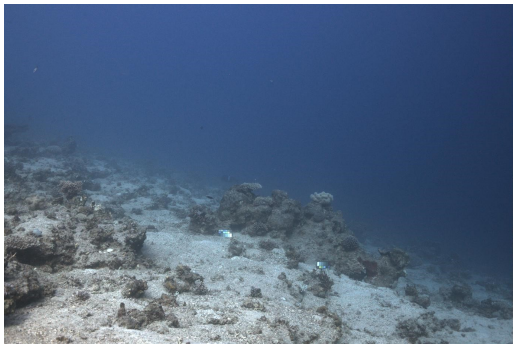
Figure 5: Reconstructed evaluation image of *"IUI3-RedSea"* scene with $\beta = 10$. Note that depths are normalized from the closest object to the furthest object or the far field if there are unbounded areas.

While the reconstructed image 5b seems visually accurate, the model wrongly perceives the water in the upper half of the image as an opaque object. This is shown in the accumulation map 5e, which shows the accumulated object weights, $\sum_{i=1}^{N} w_i^{\text{obj}}$, along a ray reduced to the corresponding pixel. The red colour in the whole image, indicative of a high accumulation, suggests that the model detects an object at some point along all rays of the image, i.e. $w_i^{obj} = 1$ for some i along each ray.[17] This observation is supported by the clear scene 5c and backscatter

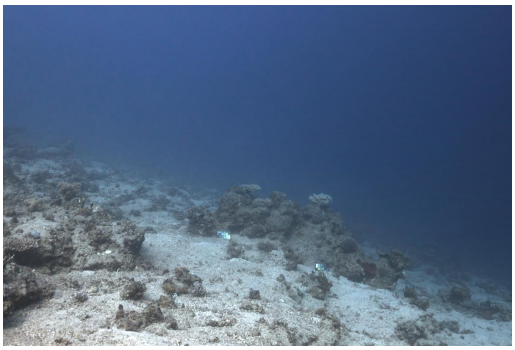
[17] To be precise, w_i^{obj} will never exactly reach one, as it is defined as $w_i^{obj} = T_i^{\text{obj}} \cdot \left(1 - \exp(-\sigma_i^{\text{obj}} \delta_i)\right)$, but only approach it asymptotically.

5d images. The clear scene incorrectly incorporates the blue water tint, while the backscatter remains absent.

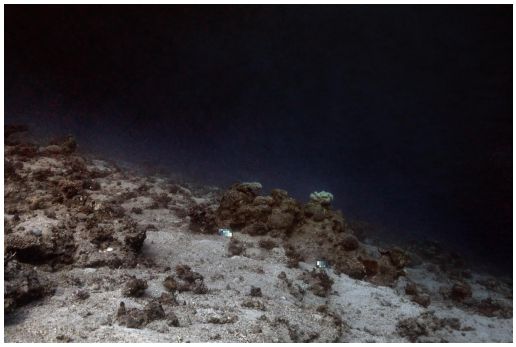
In contrast, Figure 6 shows the reconstruction results of the same evaluation image as above for a model trained with $\beta = 100$. When analyzing the object weight accumulation map 6e, it becomes evident that the model no longer perceives an object in the upper half of the image. Additionally, the clear scene reconstruction 6c does not feature any blue tint whereas the backscatter 6d is now correctly perceived as blue.



(a) Ground truth image.

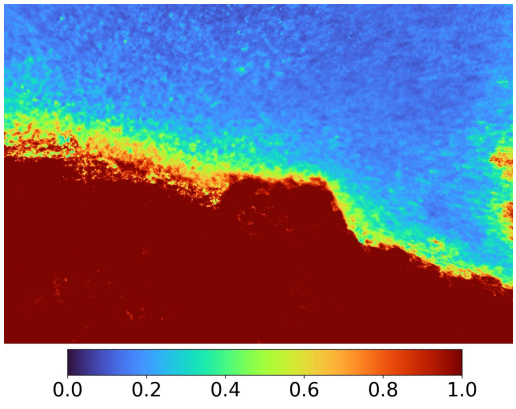


(b) Reconstructed image.

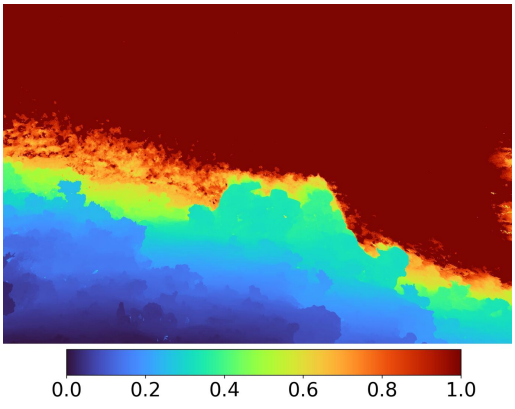


(c) Clear image.

(d) Backscatter.



(e) Object weight accumulation.



(f) Depth.

Figure 6: Reconstructed evaluation image of *"IUI3-RedSea"* scene with $\beta = 100$. Note that depths are normalized from the closest object to the furthest object or the far field if there are unbounded areas.

This comparison outlines the importance of the hyperparameter β. When opting for even larger values, the model shows deficiencies by wrongly modelling some areas with objects as semi- or fully-transparent. Moreover, the optimal value is scene dependent. Some scenes show a

correct separation for a broader range of values than others. The empirical analysis determined that a β value of 100 yields optimal performance across all four scenes within the examined dataset.

5.2 Network Design

Two other insightful sets of experiments were conducted. These delved into the configuration of the model MLPs, specifically within the object and medium networks. Additionally, different configurations of the proposal networks were examined.

The experiments showed that using two proposal networks yields the best model performance. When only one proposal network was employed, there was a notable degradation in performance metrics and visual quality. Adding more than two proposal networks did not lead to further improvements. The optimal configuration was found to involve two proposal networks, with the first one using 256 samples and the second 128 samples, while the object and medium networks only use 64 samples along each ray. Table 1 shows a comparison of the visual metrics achieved on the four scenes of the Seathru-NeRF dataset using different numbers of proposal samplers. The metrics used are common in the field, namely peak-signal-to-noise-ratio (PSNR), structural similarity index measure (SSIM) [19] and learned perceptual image patch similarity (LPIPS) [20].

Scene	**Number of Proposal Networks**								
	1			2			3		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
(1)	25.58	0.82	0.21	**27.48**	**0.93**	0.10	27.31	0.92	**0.09**
(2)	28.37	0.81	0.25	**29.13**	**0.86**	0.21	29.03	0.85	**0.20**
(3)	25.93	0.79	0.25	**30.36**	**0.93**	**0.10**	30.02	**0.93**	**0.10**
(4)	27.66	0.79	0.31	**31.74**	**0.92**	**0.14**	31.41	0.91	**0.14**

Table 1: Comparison of visual metrics on evaluation set. Scenes: (1) JapaneseGradens-RedSea [sic], (2) IUI3-RedSea, (3) Panama, (4) Curasao. **Bold font** denotes best results.

The experiments on the configuration of the model MLPs showed that increasing the layer width generally leads to improved results. However, increasing the depth of the MLPs leads to unpredictable behaviour, particularly when the water MLP has excessively many layers. In such cases, under specific hyperparameters, the model training demonstrates instabilities that finally result in exploding gradients and training divergence, despite gradient clipping.

This complex behaviour underlines the sensitivity of the model to its architectural parameters and highlights the need for careful hyperparameter tuning, especially when extending the model to new, more complex scenarios.

5.3 Comparison to Baseline

To measure the performance of the implemented method, I compare it to the official SeaThru-NeRF implementation [5] on training time and visual quality. Again, the visual metrics used for comparison are PSNR, SSIM [19] and LPIPS [20]. Table 2 shows the results on the four scenes of the SeaThru-NeRF dataset.

It is important to note that this is not a direct comparison since the authors of [5] do not disclose specific details regarding their train and test split of the images. Nonetheless, this comparison provides a general indication that the performance of my implemented model is on par, if not superior, to the official SeaThru-NeRF [5].

Scene	Training Time	This Work			SeaThru-NeRF [5] [18]		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
(1)	3h 6min	**27.48**	**0.93**	**0.10**	21.83	0.77	0.25
(2)	3h 5min	29.13	0.86	0.21	-	-	-
(3)	3h 12min	**30.36**	**0.93**	**0.10**	27.89	0.83	0.22
(4)	3h 15min	**31.74**	**0.92**	**0.14**	30.48	0.87	0.20

Table 2: Training time of implemented model and comparison of average metrics on evaluation set. Training times are measured on Nvidia A100 40GB instances and averaged over 5 runs. Scenes: (1) JapaneseGradens-RedSea [sic], (2) IUI3-RedSea, (3) Panama, (4) Curasao. **Bold font** denotes best results.

The authors of [5] provide an approximate report, indicating their model taking 10 hours on an Nvidia A100 GPU, given it runs for 250,000 iterations with a batch size of 16384. With the same batch size and number of iterations, my implementation completes in approximately 8 hours, achieving a 20% training time reduction.

However, it is important to underline that it is not necessary to train my model for such an extensive number of iterations. Convergence is observed to occur at significantly earlier stages of training. The results presented in Table 2 were obtained after 100,000 iterations, as the specified evaluation metrics stagnate beyond this point. My model takes around 3 hours to train for that amount of iterations. This presents approximately a 68% reduction in training time compared to [5].

6 Domain Applicability

This section focuses on the domain applicability of the implemented model. Together with the insights from Section 5 and synthetic datasets, it outlines the strengths and weaknesses for the proposed use case.

To obtain data that is comparable to subsea maintenance scenes, I added synthetic water effects to openly available datasets that are more representative. Those scenes are the *"dozer"* and *"plane"* scenes from the nerfstudio dataset,[19] a wind turbine inspection dataset [21] and a machine hall dataset [22].

I extended the functionality of the nerfstudio library to enable the rendering of numerical depth maps from a trained model. Subsequently, I trained an open source model, referred to as *nerfacto-huge*. This model incorporates various advancements in the field of NeRFs and is designed to excel in real-world scenes.[20] Following the training process, I extracted depth maps from the model and used them to add synthetic water effects to the corresponding input dataset via Equation (5). I adopted the parameters from [5]: specifically, $B^{\infty} = [0.07, 0.2, 0.39]$, $\beta^B = [0.95, 0.85, 0.7]$, and $\beta^D = [1.3, 1.2, 0.9]$.

While this is in theory an elegant approach to creating application-relevant scenes, it comes with an inherent problem. Namely, this approach is limited by the accuracy of the extracted depth maps. Figure 7 shows example images from the machine hall dataset with synthetic water added on top.

The synthetic subsea-like environment overall provides a visually convincing representation,

[18] Numbers are not self-computed, but taken from Table 2 of [5].

[19] Available on `https://docs.nerf.studio/en/latest/index.html`.

[20] For a comprehensive overview of the model, please see: `https://docs.nerf.studio/en/latest/nerfology/methods/nerfacto.html`

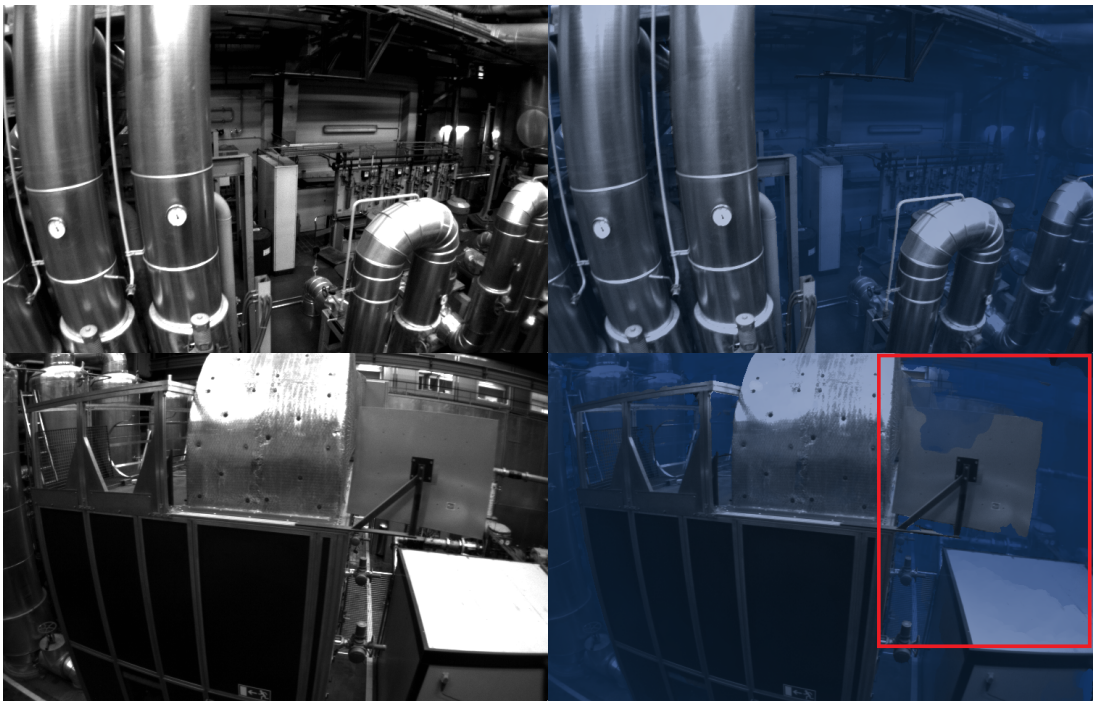


Figure 7: Original images (left) with corresponding synthetic subsea images (right). The red square highlights areas where the synthetic image shows some artefacts that are caused by inaccurate depth maps.

resembling a subsea tank system that is perceptually plausible. However, as highlighted by the red square in Figure 7, there are areas in the synthetic image where the method fails because the extracted depth maps are not accurate enough.

Figure 8 presents the results of the trained model on an evaluation image of this scene. The model correctly perceives an object everywhere in the scene as indicated by the object weight accumulation map 8e. This result is desired as the original scene is bounded by an object everywhere and there should be no areas of only medium. The reconstructed image 8b also presents a visually convincing recreation of the scene.

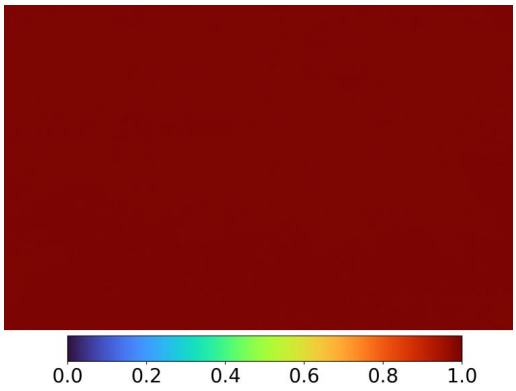
Despite this, some drawbacks still persist. First of all, the image of the clear scene 8c still retains a blue hue from the water effect, whereas the backscatter component 8d appears to be overly dark and lacks the expected blue tint. Furthermore, the rendered depthmap 8f exhibits noticeable inaccuracies. These inaccuracies are critical and must be addressed, especially if precise geometries want to be extracted down the line.

(a) Ground truth image.

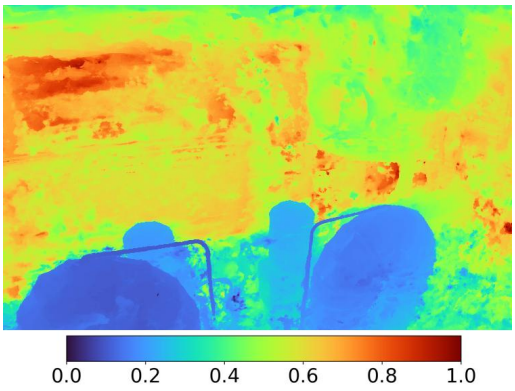
(b) Reconstructed image.

(c) Clear image.

(d) Backscatter.



(e) Object weight accumulation.



(f) Depth.

Figure 8: Reconstructed evaluation image of a synthetic machine hall evaluation image. Note that depths are normalized from the closest object to the furthest object or the far field if there are unbounded areas.

7 Discussion

Throughout this Independent Research Project (IRP), numerous challenges were encountered in trying to adapt NeRFs to the subsea maintenance domain. Some of those, such as acquiring company and application-relevant datasets, were anticipated and identified as risks in the project plan already. Others however surfaced only as the project progressed. The most significant one was the difficulty of implementing a research paper based on only the publication that lacked clarity and details.

Despite these hurdles, the final implementation provides major progress and valuable insights. First of all, through extension of the baseline approach, it provides significant improvements in training speed ($\sim$68% decrease in training time) whilst performing on par, if not better, compared to [5] (see Table 2). This is achieved by implementing a slightly modified architecture, a

different encoding strategy and a more versatile scene contraction.

The achieved visual metrics are also potentially better than those produced by SfM and MVS, but this quantitative comparison is out of the scope of this project and is considered future work.

The increase in efficiency compared to [5] is key, as it is a step towards fast training and inference which could eventually lead to real-time rendering. This advancement would be very useful to the domain as it could allow for real-time 3D models and even deployment on edge. This decrease in training time was achieved through faster MLPs. Another route to explore in the future could be to introduce other neural networks with online learning capabilities to the domain of NeRFs in order to achieve real-time reconstruction.

Furthermore, Section 5 highlights important hyperparameters of the approach and interprets them in depth. This is a crucial advancement towards understanding the intricacies of the approach and is valuable for future research and potential commercial employment.

However, the implemented approach still has certain limitations. One of those is that the optimal hyperparameters of the model are scene-dependent, as outlined in Section 5, and might not work for other scenes. This potential lack of generalisability needs to be investigated in future research and development.

Furthermore, in comparison to the results obtained from the SeaThru-NeRF dataset visualized in Figure 6, the results on the synthetic scenes presented in Figure 8 do not appear as clear and satisfying. This discrepancy indicates that either the model itself or the process of adding synthetic water effects has to be improved, possibly both. A crucial limiting factor to further investigate this was the lack of real-world 360-degree datasets, as outlined above.

8 Conclusion

This work successfully extends a subsea-specific NeRF approach to meet the demands of the subsea maintenance domain. A key achievement of the resulting implementation is a 68% reduction in training time, addressing economic constraints like the scarcity of computational resources. Furthermore, it attempts to meet the technical requirement of being able to process scenes captured from 360-degree angles by introducing a modified scene contraction.

The report fills a notable gap in existing literature by offering an in-depth analysis of critical hyperparameters affecting the model's performance. It outlines the necessity to further refine the model for increased robustness and generalizability. Moreover, it underscores the need for real-world 360-degree subsea scenes, ideally of structures relevant to the maintenance domain, that are not yet openly available. Those are crucial for further testing and evaluation of the model's capabilities.

For future research and development endeavours, it will be crucial to distil important advances from unnecessary noise in the NeRF community for this area of application. I can imagine that it could be worth introducing robust loss terms from [23] to subsea NeRFs. This might improve model generalisability and convergence as well as help with artefacts arising from moving objects in the scene such as fish.

References

[1] R. Bogue, "Robots in the offshore oil and gas industries: A review of recent developments," *Industrial Robot: the international journal of robotics research and application*, vol. 47, no. 1, pp. 1–6, 2020.

[2] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," *CVPR*, 2022.

[3] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, 102:1–102:15, Jul. 2022. DOI: 10.1145/3528223.3530127.

[4] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, *Zip-nerf: Anti-aliased grid-based neural radiance fields*, 2023. arXiv: 2304.06706 [cs.CV].

[5] D. Levy *et al.*, *Seathru-nerf: Neural radiance fields in scattering media*, 2023. arXiv: 2304.07743 [cs.CV].

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[7] A. Tewari *et al.*, "Advances in neural rendering," *Computer Graphics Forum*, vol. 41, no. 2, pp. 703–735, DOI: https://doi.org/10.1111/cgf.14507. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14507. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14507.

[8] B. Mildenhall *et al.*, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, 2019.

[9] A. V. Sethuraman, M. S. Ramanagopal, and K. A. Skinner, *Waternerf: Neural radiance fields for underwater scenes*, 2022. arXiv: 2209.13091 [cs.RO].

[10] J. Kajiya and B. von herzen, "Ray tracing volume densities," *ACM SIGGRAPH Computer Graphics*, vol. 18, pp. 165–174, Jul. 1984. DOI: 10.1145/964965.808594.

[11] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995. DOI: 10.1109/2945.468400.

[12] D. Akkaynak and T. Treibitz, "A revised underwater image formation model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[13] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[14] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, "Nerf in the dark: High dynamic range view synthesis from noisy raw images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 16 190–16 199.

[15] D. Rebain, M. Matthews, K. M. Yi, D. Lagun, and A. Tagliasacchi, "Lolnerf: Learn from one look," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 1558–1567.

[16] J. Blinn, "A trip down the graphics pipeline: Pixel coordinates," *IEEE Computer Graphics and Applications*, vol. 11, no. 4, pp. 81–85, 1991. DOI: 10.1109/38.126885.

[17] N. Rahaman *et al.*, "On the spectral bias of neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 5301–5310.

[18] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-NeRF: Structured view-dependent appearance for neural radiance fields," *CVPR*, 2022.

[19] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[20] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2018, pp. 586–595. DOI: 10.1109/CVPR.2018.00068. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00068.

[21] A. Shihavuddin and X. Chen, *Dtu - drone inspection images of wind turbine*, English, 2018. DOI: 10.17632/hd96prn3nc.2.

[22] M. Burri *et al.*, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. DOI: 10.1177/0278364915620033. eprint: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract.

[23] S. Sabour, S. Vora, D. Duckworth, I. Krasin, D. J. Fleet, and A. Tagliasacchi, "Robustnerf: Ignoring distractors with robust losses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 20 626–20 636.

Appendices

A Weigths & Biases

Figures A.1, A.2 and A.3 provide insights into the tracking of experiments used throughout this work.

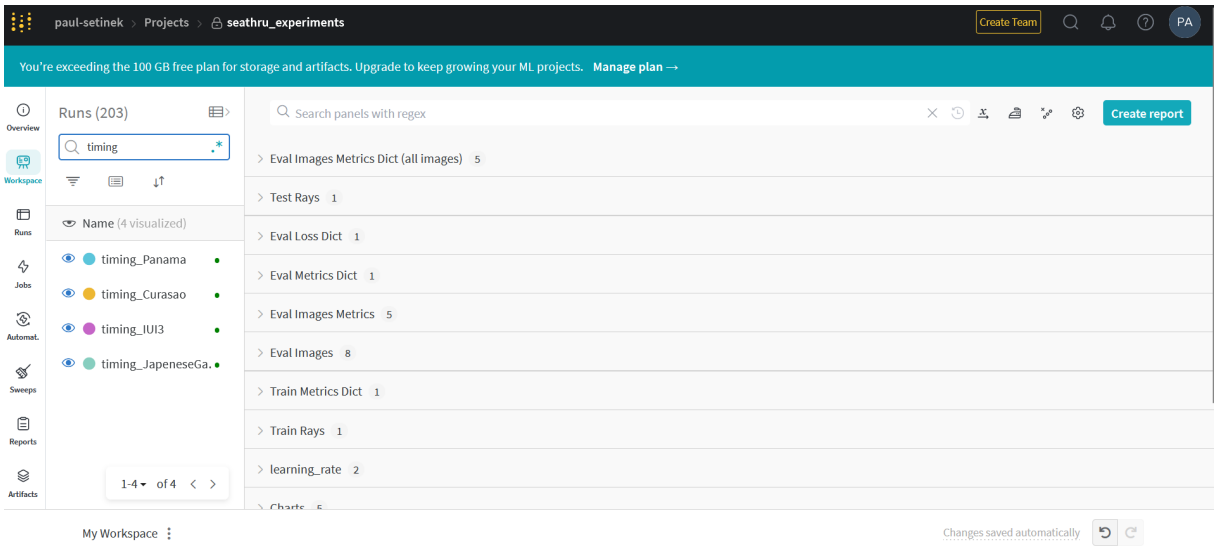


Figure A.1: Example overview of a project page in W&B. This is an example where I timed my method on the SeaThru-NeRF dataset. The different panels were used to group data to compare experiments.

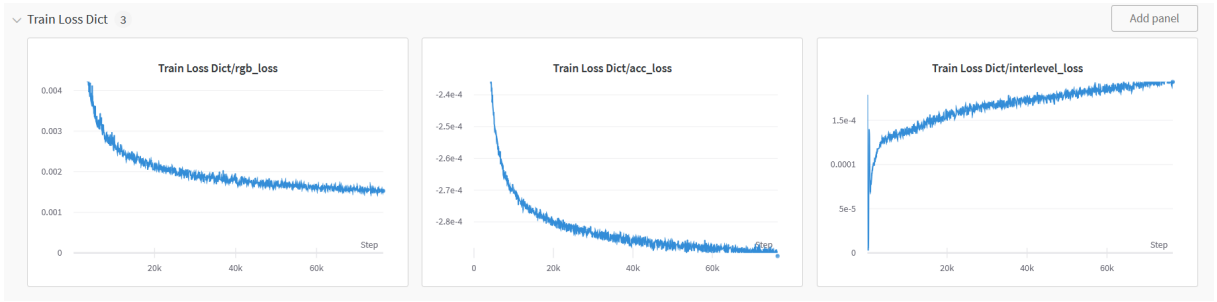


Figure A.2: Example of the panel displaying different loss components during training. Three loss components are shown as described in Section 2.3. This example shows the phenomenon of a negative acc_loss as described in Figure 4, indicating good separation of object and medium regions.

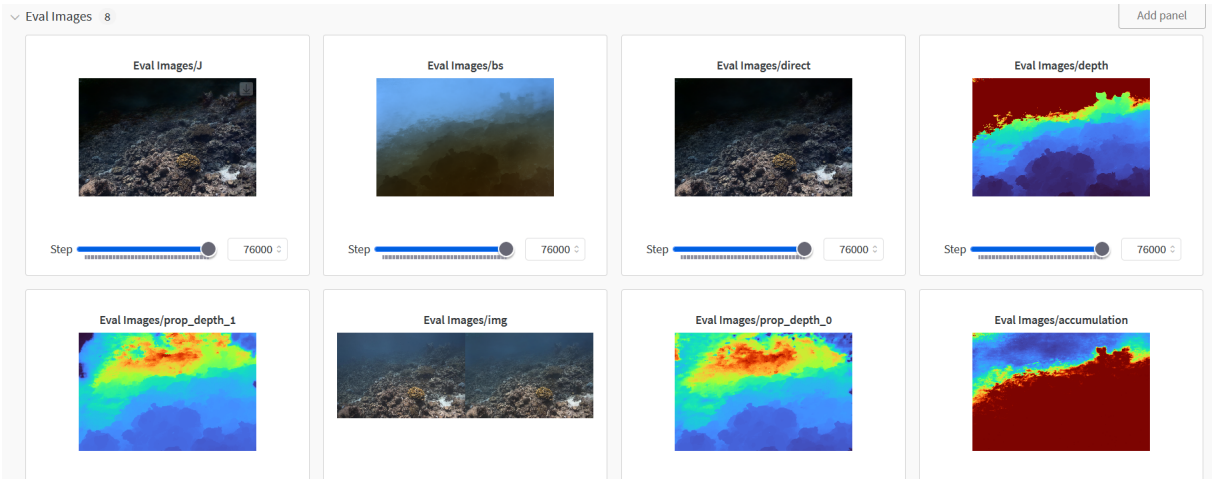


Figure A.3: Example of the panel showing renderings of an evaluation image during training. I render various evaluation images (RGB, clean, backscatter depth, object weight accumulation) every 500 iterations during training. This allows for visual comparison and increased interpretability of the model as interpreting convergence and performance of NeRFs often needs more than numerical loss values or metrics.

B Experiments Overview

Experiment Name	Description	Outcome
use_viewing_dir	Investigate the effect of using viewing direction as inputs to the object's colour MLP.	Not using it performs on par compared to using it and avoids some flickerings and instabilites.
weights_prior	Investigate whether putting the prior from Equation (12) on the object weights rather than transmittance leads to better separation and convergence.	No separation (at least not with $\beta = 100$ and $\beta = 500$). In this case the β factor needs to be in front of the other term in the prior formulation.
density_bias	Investigate the effect of introducing a density bias before the last activations to have a better initialization.	No significant changes in model convergence and results.
prop_nets	Investigate the effect of different configurations of proposal samplers.	Two proposal networks with 256 and 128 samples respectively lead to the best results.
lambda	Investigate the effect of different λ values.	$\lambda = 0.0001$ leads to the a good separation on all scenes (combined with $\beta = 100$).
lambda_decay	Investigate the effect of higher λ values at the start of training.	Does not lead to better separation, convergence or results.
MLP_size	Investigate the effects of different MLP sizes for object and medium MLP.	Generally, wider networks lead to more accurate reconstructions. Very deep water MLPs ($>$5 layers) lead to diverging training.
beta	Investigate the effect of different β factors in the prior Equation (12).	Proper choice is crucial for proper separation between object and medium. The optimal value for the investigated dataset is $\beta = 100$.

Table B.1: Summary of conducted experiments with their respective outcomes. All the experiments are logged on W&B and can be shared upon request.