Universidad Zaragoza

Master's Thesis

Real-time rendering of underwater scenes based on data and an approximation to multiple scattering

Author

Néstor Monzón González

Supervisors

Adolfo Muñoz Orbañanos

Diego Gutiérrez Pérez

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2022

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Néstor Monzón González ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de Máster Universitario en Robótica, Gráficos y Visión por Computador (Título del Trabajo)

Real-time rendering of underwater scenes based on data and an approximation to multiple scattering.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24 de noviembre de 2022

Fdo: MONZON GONZALEZ NESTOR - 18065844B

Firmado digitalmente por MONZON GONZALEZ NESTOR - 18065844B
Fecha: 2022.11.24 10:14:19 +01'00'

Contents

Abstract

We propose a physically-based, multispectral simulation to render underwater scenarios in real time, which also takes into account the RGB response curves of arbitrary sensors. Our approach can simulate the two main phenomena of underwater illumination. First, there is multiple scattering, light scattered several times between particles before reaching the sensor. This phenomenon is therefore very low frequency and can be modeled as a function of depth and wavelength. We propose a physically-based approximation to this complex phenomenon based on the measurable coefficient of diffuse downwelling attenuation.

The second key component of underwater appearance is single scattering, light refracted on the surface that is scattered once before reaching the eye. Single scattering is responsible for the volumetric shadows and light rays visible underwater, and its simulation is key for real-time visualizations. We leverage a combination of volumetric lighting and caustics projection real-time techniques to provide a plausible approximation.

We have implemented our proposed mixed method in Unity, and we show how our method is able to react to a variety of physical conditions, such as the water type or the underwater depth. We use measured data from oceanography, and show the ability of our method to approximate the appearance under different scattering, absorption and downwelling coefficients, as measured in the well known Jerlov water types from oceanography. We compare our approximation to a path traced simulation, showing great accuracy at a fraction of the computational cost. We also compare our method to another state-of-the-art method for underwater rendering, and show how our proposed model is more physically-based and controllable.

Acknowledgements

This work would not have been possible without the invaluable guidance from my supervisors Adolfo Muñoz and Diego Gutiérrez. I would like to thank them for their time and their patience on this process.

I would also like to thank everybody working in the *Graphics and Imaging Lab* for the amazing working atmosphere we share. Mercedes Fatás, the group administrative, also deserves special recognition: nothing would be possible without her tireless efforts.

Finally, I would like to thank Derya Akkaynak for her spectacular research seminar on underwater computer vision in September 2021: without it, I would not have worked on this topic. Her knowledge as a collaborator of the group afterwards has also greatly aided in the making of this project.

1. Introduction

Ocean waters are arguably one of the richest media in nature, and thus show great variety in their appearance. As water is a participating medium, light differentially interacts with water molecules, as well as water constituents such as suspended particles and organic matter. The concentrations and nature of these constituents completely alter the appearance of water bodies. Thorough understanding of ocean water appearance has multiple applications that range from oceanography, underwater imaging, remote sensing, computer vision and even biology and ecology.

However, as interactions happen at every differential point along the trajectory of light within water, its appearance is not only hard to predict but also challenging to understand. A classical option for underwater appearance prediction is to resort to light transport simulation, in which, based on the physics of light transport and on the optical properties of water and its constituents, images can be generated. However, such simulations (see section 1.1 for more detail) are prohibitively costly in terms of computation time to be of practical use for some applications. On the other end of the spectrum, most real-time implementations of underwater rendering are focused on aesthetics and artistic control, with little to no accuracy with respect to the physics of light-water interactions, and are therefore unfit for applications besides art and entertainment.

There is a need to bridge this gap: a physically-based method for real-time rendering of underwater scenarios, that is useful and practical for oceanography-related applications, which is what we propose in this Master Thesis. This is of course rather challenging: with a budget per frame of at most 34 milliseconds (for real time) it is impossible to accurately simulate the whole physics of light transport. In order to achieve that, we need approximations and low-computational-cost underwater models that are, still, based on the physics of light transport so that they are close enough to full light transport simulations.

Furthermore, besides accuracy, physically-based rendering approximations (as opposed to *ad hoc*, artistic solutions) can be of great use for even entertainment purposes. As physical models react to a number of physical parameters, modifying these parameters directly alters the resulting appearance, whereas artistic solutions need manual tuning to adapt to different conditions. For example, following physical formulas can allow the application to simulate a variety of measured waters from the literature with different turbidity and albedo (e.g., the Jerlov water types [SM15]), and react underwater depth.

Proposing such a solution is a challenging task, as oceans are a very rich optical medium with a great variety of complex light transport phenomena [GJJD09]. One key idea is that

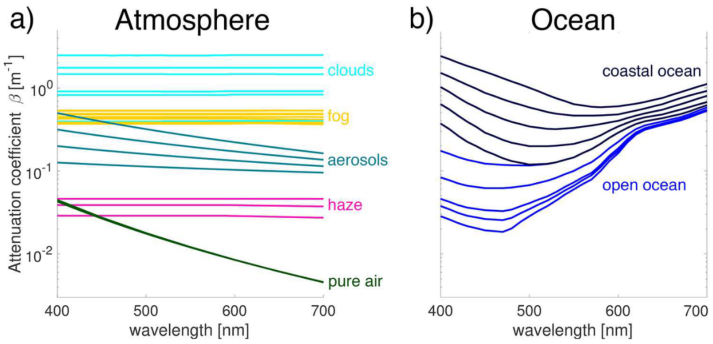


Figure 1.1: Comparison of the extinction of different media in the atmosphere (a) and in the ocean (b). Plots courtesy of Derya Akkaynak [AT18].

many of these phenomena depend directly on wavelength. Most notably, the more common atmospheric image formation model is not appropriate to simulate the appearance of the ocean, since the absorption coefficient ($\sigma_a(\lambda)$, in m^{-1}, which is the probability that light of wavelength λ is absorbed per meter) shows great wavelength dependency, and is often more prevalent than the scattering coefficient [AT18]. This results in vastly different extinction coefficients: as we can see in Figure 1.1, we can safely ignore the dependency in wavelength on the atmosphere (a), but not in the oceans (b).

Other optical phenomena present in the ocean, like inelastic scattering [GSMA08], or the complex photoreceptors of some animals like the mantis shrimp [CBMC14], can also only be modeled by taking this wavelength dependency into account. Although we do not focus on these in this work, proposing a hyper-spectral pipeline for the approximation of underwater illumination in real time is the first step towards being able to tackle them in the future.

However, while we believe that accounting for the spectral nature of light is crucial for simulating underwater appearance in a physically based way, doing so increases the computational cost of the simulation. Instead of evaluating the lighting computations for the usual three (RGB) wavelength bands, we need a higher density of wavelength samples. The implementation of hyper-spectral rendering will be further detailed in 4.2.

Our proposed solution to these challenges is our first contribution, a real-time analytical approximation to spectral multiple scattering. To achieve this, we build upon works in the field of oceanography, which have extensively studied the interaction of light with the ocean waters. We propose to model underwater ambient illumination as a function of depth inspired by real measurements, allowing to greatly reduce the cost of fully simulating multiple scattering (see chapters 3 and 4) by deriving an analytical expression which can be evaluated in constant time per wavelength (section 4.1).

On the other hand, we note that single scattering effects are especially crucial in underwater appearance, especially near the ocean surface. They are formed by the refraction and shadowing of the sun's lighting when traversing the deforming surface of the water. We propose to emulate these effects utilizing a combination of volumetric lighting approaches usually used for real-time fog, called froxel lighting [Vos14], and a moving emulation of the caustic patterns inspired by caustics mapping [Sta96]. This method is further detailed in chapter 5.

We have implemented the combination of our multiple scattering approximation and our single scattering method in Unity, resulting in our Underwater Rendering Pipeline (UWRP, see section 4.3.2 for more details). We have also built a demo application utilizing this pipeline. This application allows for basic navigation of the scene, and features a user interface to modify many of the physical parameters of the simulation in real time.

In summary, having a spectral physically-based pipeline allows the simulation to naturally react to a number of physical parameters, including depth, the sensor type, the medium type, the strength of the sun, etc.

Finally, in the results chapter (6), we show both a qualitative analysis of our pipeline, as well as a quantitative evaluation of our multiple scattering approximation when compared with a path traced solution. The results are very promising, as this approximation achieves very similar results to the reference several orders of magnitude faster, under a variety of conditions.

1.1 Related work

Light transport in ocean bodies has been studied for decades in several overlapping fields. Mainly, there is the field of oceanography, where many different works are devoted to the study of the interaction of light with the water media. There is also the closely related field of underwater computer vision. And finally, the field of computer graphics for simulating underwater light transport.

Oceanography Usually, in this context, as the ocean itself is the object of interest, radiance transfer is studied as a vehicle to gain some deeper knowledge about the water body. For example, radiance measurements at different depths can be used to estimate the concentrations of different particles in the water [RPC89; Blo19].

In this field, the study of light is also of great importance from the biological and ecological points of view, as sun radiance is the main energy source powering all life through photosynthesis.

In this context, the real world phenomena we are trying to emulate have been modeled and studied for decades, providing a strong theoretical foundation for our work. Most notably, our multiple scattering approximation is based on the diffuse downwelling attenuation coefficient Kd [Mob94; LDC*05] (see chapter 4 for an in-depth explanation). Kd is an empirically measured coefficient on different water bodies, and although it is not as invariable as the Inherent Optical Properties (absorption and scattering coefficients) [Mob01], it is considered a specially descriptive and invariant Apparent Optical Property [Gor89]. This is why it is part of the measured coefficients used to classify the well known Jerlov water types [SM15], and why we believe it is a strong foundation for our approximation.

Underwater computer vision Underwater computer vision is a growing field closely related to oceanography. There, the main interest is usually to remove the interaction of light with the medium in order to recover higher quality images for further processing [SK04]. This is the problem of underwater image dehazing, which differs a lot from atmospheric dehazing due to

fundamental differences in the medium [AT18], mainly, the strong wavelength dependency of the extinction coefficient we have seen in Figure 1.1.

Simulation of participating media In computer graphics, the goal is closest to ours: simulating the appearance of the ocean or, more generally, participating media. However, there can be two possible secondary goals: artistic freedom (movies, games) and physical accuracy. The most physically based methods are often offline: they focus on accuracy instead of performance.

As rendering participating media is a fundamentally complex problem, many different works have studied ways to tackle it. In general, all works focus on solving or approximating the Radiative Transfer Equation, introduced by Chandrasekhar [Cha50], which is a differential model of the behaviour of all participating media. We will explain the problem in more detail in section 2, but the main takeaway is that all offline and real-time methods are solving this mathematical problem, chosing each a set of tradeoffs between accuracy and speed.

The most basic of these offline methods is Path Tracing, where Monte Carlo integration is used to approximate the illumination in the scene [Kaj86; AK90]. Monte Carlo integration ensures an *unbiased* approximation of the illumination, meaning the result would converge (reduce the noise of the output) provided infinite time. As we do not usually have infinite time, some works improve the convergence speed by better sampling or zero-variance methods [PM93; GKH*13; Kd14; MHD16]. Others focus on better searches of the illumination sources, with methods such as Bidirectional Path Tracing [LW96; JA18], Metropolis Light Transport [VG97; PKK00], or Path Guiding [HZE*19; DWWH20]. Recently, neural approaches have also been proposed in this field [MGN17; MMR*19].

Concurrently, Photon Mapping approaches were developed, which model the illumination by first caching the illumination of the sources in photon maps [Jen95; JC98]. These were extended into Photon Beams [JZJ08; JNSJ11; JNT*11; KGH*14], which improve their performance and accuracy. A conceptually similar approach is radiance caching [JDZJ08; MJJG18], which stores radiance in efficient structures.

Offline ocean rendering Some methods have further specialized on oceanic light transport. This is a very rich field due to the large number of phenomena involved. Many works focus on the dynamic aspect of the ocean like simulating the water surface, either for aesthetic purposes, or in more physically based approximations [KTT13; NSB13; LX13; JW15; JW17; PP20]. As we focus on underwater scenarios, we do not need to fully simulate the water surface.

One of these works focusing on surfaces used a similar approximation to ours to compute the radiance resulting from subsurface scattering in the water body depending on its physical parameters [PA00]. However, they do not focus on full underwater lighting, and their method is offline, allowing a higher cost than ours.

Others, such as Gutierrez et al. [GSMA08], focus on proposing more extensive bio-optical ocean models to account for usually ignored phenomena like inelastic scattering. As we mentioned in the introduction, although we do not account for these events, our proposed spectral pipeline is the first step towards being able to model these phenomena in real time.

Real-time ocean rendering Few works are devoted to the underwater light transport in real time. In game engines, artistic control is often more important than physical accuracy. For example, in Subnautica (2018), which takes place underwater, earlier prototypes of the rendering system followed physically based rendering, but it had to be changed to a more artistic style to make it more varied and improve visibility [Dev16].

Typical game engines not focused on underwater scenarios, like Unity, just apply a depth-dependant color shift to the image.

A typical approach in interactive applications is to consider the ambient light dependant on depth, and a blue tinted fog to emulate the water. One example is Liarokapis et al. [LKA*17], who follow such an approach to render underwater interactive archaeology sites in VR. Their work is used to visualize underwater archaeological assets processed through photogrammetry, and propose several additions to increase the immersion in the application, such as placing procedural assets, animating fish, etc.

However, their lighting is not physically based at all, considering ambient light intensity dependant on depth with a heuristic to change its color and applying linear interpolation in between. Although it works for this use case, this does not allow to simulate different water media through the measured coefficients as we do, and much less render the scene hyperspectrally.

Other methods use Image Based Lighting (IBL), which obtains the ambient light from an environment map (an image fully surrounding the scene). This results in realistic and varying lighting for small scenes, as every direction of the surroundings can have a different radiance. Some methods apply this to underwater lighting. For example, Thompson et al. [TCR19] proposes a pipeline to realistically blend a virtual object into a 360 degrees underwater video in real time. Their IBL based lighting achieves better blending than more naive approaches (such as the previous one) according to their user study.

They also test the effect of adding:

- Caustics on the virtual objects.
- God rays: volumetric caustics, produced by the scattering of the caustics on the water particles.
- Floating particulates on the water (simulating macroscopic small impurities usually visible underwater).

They find that, generally, particulates tend to break immersion, but caustics and God Rays improve the appearance of realism. Their caustics are rendered according to the estimated position of the sun on the original video. IBL methods have their use cases, and mixed reality blending is a perfect example, but they are limited in their controllability and flexibility for other uses.

In contrast, a more physically based approach, such as our proposed method, as it only depends on physical parameters, can simulate a wider range of scenarios. Furthermore, we do not work on RGB, and we do not use an atmospheric fog (with monochrome absorption). Our method is also able to render caustics (on surfaces and volumetric).

USSim [CZSZ22] is the closest work to our method. Their objective is a controllable (i.e., that organically reacts to physical parameters such as underwater depth) and physically based model for underwater lighting. They make use of a similar depth-dependant irradiance approximation used in oceanography for ambient illumination. However, they approximate lighting using the wideband (RGB) attenuation coefficients for efficiency. In many cases, especially in absorption-dominated waters, this can lead to errors [AT18]. Our proposed solution is more costly, as it computes the radiance of more wavelength bands, but it should perform better in absorption-dominated waters. Furthermore, our method is more general, as we also tackle single scattering and caustics, whereas they only focus on the ambient term.

2. Theoretical background

Before we can delve deeper into our proposed method, we will introduce the problem in a more formal way. To summarize it, rendering participating media, such as water, is a fundamentally harder problem than just rendering surfaces (assuming no interaction as the light traverses space). In the following sections, we will detail the mathematical reasons behind this.

Shading surfaces in computer graphics is a well studied problem. For a given pixel in the image to be synthesized, we know its position in the world $\mathbf{x}$ and the direction of the ray $\boldsymbol{\omega}_o$ which travels towards the eye. Its color is physically the radiance of that ray L_s, which we can compute using the rendering equation [Kaj86]:

$$L_s(\mathbf{x}, \boldsymbol{\omega}_o) = L_e(\mathbf{x}, \boldsymbol{\omega}_o) + \int_\Omega f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_i)(\boldsymbol{\omega}_i \cdot \mathbf{n})\, d\boldsymbol{\omega}_i, \tag{2.1}$$

where L_e is the emission of the surface. The integral term represents the reflected light, adding together the incident radiance from every direction (directions $\boldsymbol{\omega}_i$ belonging to the sphere Ω centered at point $\mathbf{x}$). The incident radiance from a given direction is L_i, which is weighted by the bidirectional scattering reflection function (or BSDF) f_r. Finally, the cosine term $(\boldsymbol{\omega}_i \cdot \mathbf{n})$ accounts for the geometric foreshortening of the angle with the surface normal $\mathbf{n}$.

Even though computing the integral can be very costly, multiple methods have been developed for decades to approximate it with the best possible quality while reducing the cost. There are offline methods, such as path tracing, which offer great quality with a large computational cost, based on Monte Carlo integration. In real time, less expensive approximations are required, and often rasterization is used instead of ray tracing.

2.1 Radiative Transfer Equation

However, the previous integral assumes that the beam of light does not interact with the medium between the surface and the eye. In participating media, such as water, fog or smoke, every point along the ray contributes towards the final radiance. This is because the medium is composed of particles that may emit light, scatter it (reflect in any direction) or absorb it (physically, its electromagnetic energy is transformed in other forms).

The Radiative Transfer Equation, or RTE, introduced by Chandrasekhar in 1950 [Cha50], describes these interactions of light with media in a differential manner according to only three

coefficients:

- The scattering coefficient, σ_s (m^{-1}), the differential probability of light being scattered per unit length travelled along the medium.
- The absorption coefficient σ_a (m^{-1}), the probability of absorption.
- The emission coefficient σ_e (m^{-1}), the probability of emission. As in this work we are focusing on water media, which are of course non-emissive, we can ignore this term altogether.

The RTE defines the difference in radiance for a ray with direction $\boldsymbol{\omega}$, at point $\mathbf{x_z}$, advancing a differential length dz through the medium as in Figure 2.1. This difference in radiance $\frac{\mathrm{d}L(\mathbf{x_z},\boldsymbol{\omega})}{\mathrm{d}z}$, in a non-emissive medium, will be:

$$\frac{\mathrm{d}L(\mathbf{x_z},\boldsymbol{\omega})}{\mathrm{d}z} = \sigma_s L_i(\mathbf{x_z},\boldsymbol{\omega}) - \sigma_t L(\mathbf{x_z},\boldsymbol{\omega}). \tag{2.2}$$

Radiance can be gained due to in-scattering $\sigma_s L_i(\mathbf{x_z},\boldsymbol{\omega})$ (Figure 2.2a), which is light coming from any direction $(L_i(\mathbf{x_z},\boldsymbol{\omega}))$ scattered towards $\boldsymbol{\omega}$. The lost radiance results from the extinction coefficient $\sigma_t = \sigma_a + \sigma_s$, which accounts for both out-scattering (Figure 2.2b, radiance that would have reached the eye but is scattered in other directions) and absorption (Figure 2.2c, radiance converted to other forms of energy).

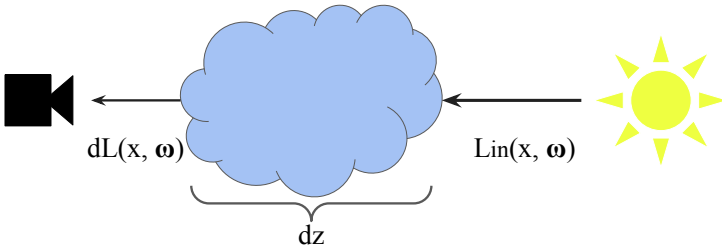
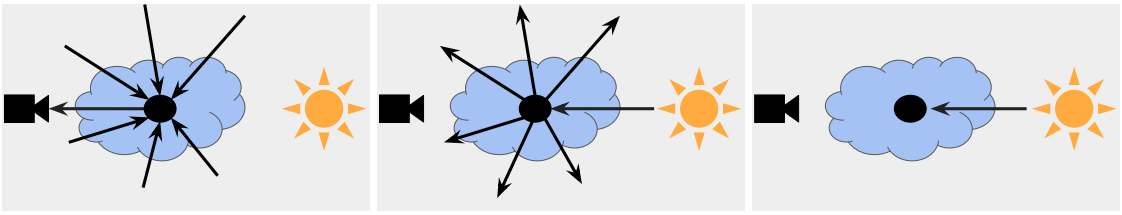


Figure 2.1: Difference in radiance for ray stepping through a medium.



(a) In-scattering (b) Out-scattering (c) Absorption.

Figure 2.2: All possible events in a non-emissive medium. In-scattering adds radiance, while out-scattering and absorption *extinguish* it.

In more detail, the in-scattering term $L_i(\mathbf{x},\boldsymbol{\omega})$ needs to account for radiance from every direction:

$$L_i(\mathbf{x}, \boldsymbol{\omega}) = \int_\Omega f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}) L(\mathbf{x}, \boldsymbol{\omega}_i)\, d\boldsymbol{\omega}_i, \tag{2.3}$$

where Ω represents the set of all directions in the unit sphere, $L(\mathbf{x}, \boldsymbol{\omega}_i)$ is the incident radiance on a given direction $\boldsymbol{\omega}_i$, and $f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega})$ is the phase function of the medium. Phase functions are the medium equivalent of BSDFs for surfaces: the ratio of light incident at point x with direction $\boldsymbol{\omega}_i$ which is scattered in direction $\boldsymbol{\omega}$.

Note that this integral over the incident radiance will need to be accounted for at every point in the medium, hinting at the computational complexity of the volume rendering problem as opposed to surface rendering.

In isotropic media, where scattered light is distributed uniformly in the sphere, the value of the phase function is constant, $f_s(x, \boldsymbol{\omega}_i, \boldsymbol{\omega}) = \frac{1}{4\pi}$. This is derived from the inverse of the area of the sphere. Water is not typically isotropic, but we approximate it this way to allow for a more efficient approximation, as we will see in chapter 4. In this case,

$$L_i(\mathbf{x}, \boldsymbol{\omega}) = \frac{\sigma_s}{4\pi} \int_\Omega L(\mathbf{x}, \boldsymbol{\omega}_i)\, d\boldsymbol{\omega}_i. \tag{2.4}$$

This results in the following expression for the RTE of our non-emissive, isotropic medium:

$$\frac{\mathrm{d}L(x_z, \boldsymbol{\omega})}{\mathrm{d}z} = -\sigma_t L(x_z, \boldsymbol{\omega}) + \frac{\sigma_s}{4\pi} \int_\Omega L(\mathbf{x}, \boldsymbol{\omega}_i)\, d\boldsymbol{\omega}_i. \tag{2.5}$$

2.2 Volume rendering equation

Although there have been attempts to solve the RTE in its differential form in other ways [Muñ14], usually it is useful to re-frame the problem and integrate this function in order to compute the medium contribution for a ray of a given length. To recap, the radiance we need to compute is shown in Figure 2.3, for a given ray coming from the surface of a submerged object at point $\mathbf{x_S}$, in direction $\boldsymbol{\omega}$, towards the camera (at point x_0, S units away).

We already know the radiance coming from the surface, $L_s(\mathbf{x_S}, \boldsymbol{\omega})$, thanks to the rendering equation (2.1). For any given point $\mathbf{x_z}$ along the ray, we know its contribution from in-scattering ($L_i(\mathbf{x_z}, \boldsymbol{\omega})$) from the RTE. As every point along the ray must be accounted for, this results in an integral along the ray $\int_0^S L_i(\mathbf{x_z}, \boldsymbol{\omega})\, dz$.

However, this would only add radiance, and we know that the medium extinguishes incoming radiance due to absorption and out-scattering (RTE, Equation (2.5)). This is solved by defining transmittance $T(x_z)$, the ratio of radiance lost between point x_z ($L(\mathbf{x_z}, \boldsymbol{\omega})$) and the sensor ($L(\mathbf{x_0}, \boldsymbol{\omega})$),

$$T(\mathbf{x_z}) = \frac{L(x_0, \boldsymbol{\omega})}{L(x_z, \boldsymbol{\omega})}, \tag{2.6}$$

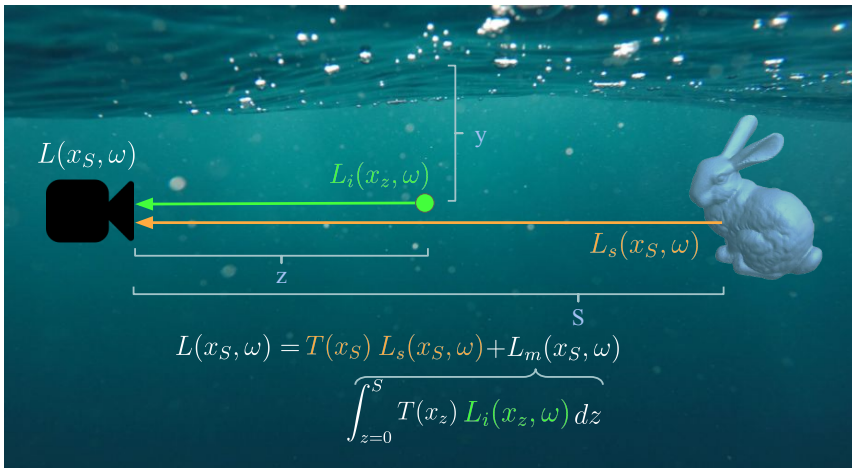


Figure 2.3: The final radiance L along the ray $\boldsymbol{\omega}$ depends on the radiance from the object L_s, the one from the medium L_m (composed of the in-scattered radiance L_i of every differential point along its path), and the attenuation T (Background from Jeremy Bishop in Unsplash).

we can now rewrite the RTE in its integral form, which is usually referred to as the Volume Rendering Equation or VRE:

$$L(\mathbf{x_S}, \boldsymbol{\omega}) = T(\mathbf{x_S}) L_s(\mathbf{x_S}, \boldsymbol{\omega}) + \int_0^S T(\mathbf{x_z}) L_i(\mathbf{x_z}, \boldsymbol{\omega})\, dz. \tag{2.7}$$

Transmittance To find the transmittance of a medium between points $\mathbf{x_0}$ and $\mathbf{x_s}$, at distance S, we can use the previous differential definition for the extinction, $\frac{\mathrm{d}L(\mathbf{x},\boldsymbol{\omega})}{\mathrm{d}z} = -\sigma_t L(\mathbf{x}, \boldsymbol{\omega})$, and solve the resulting ordinary differential equation:

$$\begin{aligned}
\frac{\mathrm{d}L(\mathbf{x}, \boldsymbol{\omega})}{\mathrm{d}z} &= -\sigma_t L(\mathbf{x}, \boldsymbol{\omega}) \\
\frac{1}{L(\mathbf{x}, \boldsymbol{\omega})} dL(x, \boldsymbol{\omega}) &= -\sigma_t\, dz \\
\int_0^S \frac{1}{L(\mathbf{x}, \boldsymbol{\omega})} dL(x, \boldsymbol{\omega}) &= \int_0^S -\sigma_t\, dz \\
ln(L(x_s, \boldsymbol{\omega})) - ln(L(x_0, \boldsymbol{\omega})) &= -\int_0^S \sigma_t\, dz. \\
ln\left(\frac{L(x_s, \boldsymbol{\omega})}{L(x_0, \boldsymbol{\omega})}\right) &= -\int_0^S \sigma_t\, dz.
\end{aligned} \tag{2.8}$$

As $T(\mathbf{x_s}) = \frac{L(x_s,\boldsymbol{\omega})}{L(x_0,\boldsymbol{\omega})}$, we now only need to take the exponential of both sides:

$$T(\mathbf{x_s}) = \frac{L(x_s, \boldsymbol{\omega})}{L(x_0, \boldsymbol{\omega})} = e^{-\int_0^S \sigma_t\, dz}. \tag{2.9}$$

Note that, in a heterogeneous medium, where there is a spatial variation on the constituents (organic or inorganic particles) that define its optical properties, the value of σ_t would depend on the point x. However, if we assume the medium is homogeneous (the constituents do not depend on the position), which is a good approximation for water volumes, its value is constant and the integral is just:

$$T(\mathbf{x_s}) = e^{-\int_0^S \sigma_t \, dz} = e^{-S\sigma_t}. \tag{2.10}$$

Integrated in-scattered radiance The in-scattered radiance from the medium at point $\mathbf{x}$, $L_i(\mathbf{x}, \boldsymbol{\omega})$, as seen in Figure 2.3, corresponds to the in-scattering term in the RTE, from Equation (2.3). This results in the following expanded version of the VRE:

$$\begin{aligned} L(\mathbf{x_S}, \boldsymbol{\omega}) &= T(\mathbf{x_S}) L_s(\mathbf{x_S}, \boldsymbol{\omega}) + \int_0^S T(\mathbf{x_z}) L_i(\mathbf{x_z}, \boldsymbol{\omega}) \, dz \\ &= T(\mathbf{x_S}) L_s(\mathbf{x_S}, \boldsymbol{\omega}) + \int_0^S \left(T(\mathbf{x_z}) \frac{\sigma_s}{4\pi} \int_\Omega L(\mathbf{x}, \boldsymbol{\omega}_i) \, d\boldsymbol{\omega}_i \right) dz. \end{aligned} \tag{2.11}$$

As we can see, for every ray, we not only need to compute the integral along its length of the in-scattered radiance at that point. This radiance also requires a second integral around the sphere. We propose a two-part method to approximate this integral to reduce the computational cost of fully simulating it, enabling a real time visualization, while remaining physically based.

3. Multiple and single scattering

In-scattering events, responsible for the radiance at a given point $\mathbf{x_z}$ in the medium ($L_i(\mathbf{x_z}, \boldsymbol{\omega})$), are often classified in computer graphics as single scattering and multiple scattering, as shown in Figure 3.1. Single scattering (in orange in the figure) is light that is scattered directly from the light source (the water surface in our case, with radiance $L_{WS}(\mathbf{x}, \boldsymbol{\omega}_i)$) towards the sensor.

Multiple scattering, on the other hand, occurs when several scattering events are chained before reaching the sensor (in yellow in Figure 3.1). Separating these two events may seem arbitrary, but it is useful when it comes to simulate scattering, as each show significant differences, requiring different techniques to efficiently model them.

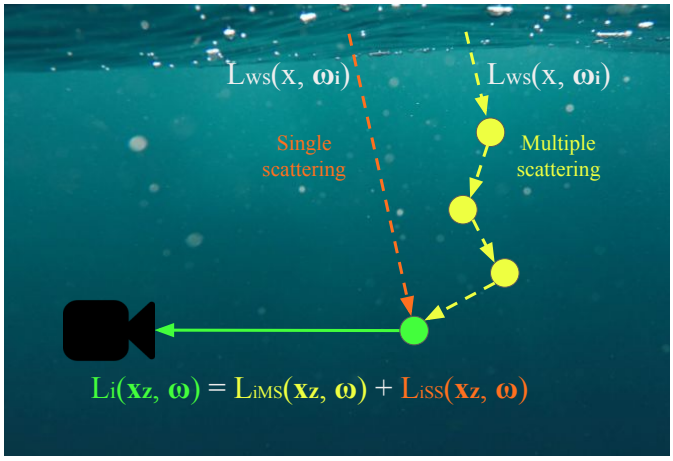
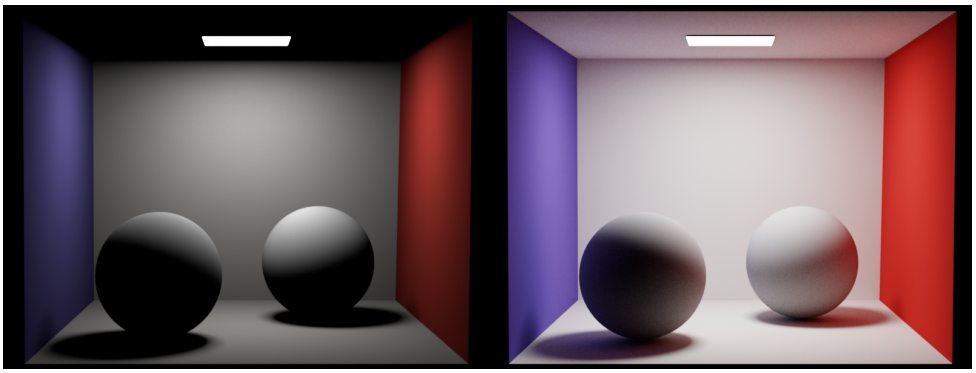


Figure 3.1: Orange: single scattering, light is scattered directly from the source ($L_{WS}(\mathbf{x}, \boldsymbol{\omega}_i)$) towards the eye. Yellow: Multiple scattering, radiance from the water surface scatters multiple times between the particles before reaching the eye. The total in-scattering (green) at that point will be the addition of the two: $L_i(\mathbf{x_z}, \boldsymbol{\omega}) = L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega}) + L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega})$

For example, multiple scattering illumination shows lower spatial variance, as considering multiple indirect light bounces acts as a statistical filter, effectively smoothing out the illumination. This is analogous to direct illumination as compared to indirect illumination in standard path tracers. In Figure 3.2, we can see how considering indirect lighting (right) brings much lower frequency illumination than direct lighting (left), which only shows very hard shadows.

Due to these differences, we propose a *mixed pipeline* to render both low-frequency multiple scattering as well as high frequency, temporally varying single scattering. Most importantly, this separation is physically correct: if we recall the medium term from the VRE (Equation (2.7)),



(a) Direct lighting (b) Direct and indirect lighting

Figure 3.2: Comparison of direct lighting and indirect lighting on a path traced scene. Indirect lighting shows lower frequency spatial variation due to its statistical nature. This is analogous to multiple scattering as opposed to single scattering.

$$L_m(\mathbf{x_S}, \boldsymbol{\omega}) = \int_0^S T(\mathbf{x_z}) L_i(\mathbf{x_z}, \boldsymbol{\omega})\, dz, \tag{3.1}$$

the in-scattered radiance $L_i(\mathbf{x_z}, \boldsymbol{\omega})$ at a given point can be decomposed into the radiance resulting from multiple scattering events $L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega})$, and the one from single scattering $L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega})$: $L_i(\mathbf{x_z}, \boldsymbol{\omega}) = L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega}) + L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega})$. Performing this substitution in Equation (3.1), we obtain the following expression, easily separated into two integrals through the associative property:

$$\begin{aligned} L_m(\mathbf{x_S}, \boldsymbol{\omega}) &= \int_0^S T(\mathbf{x_z})(L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega}) + L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega}))\, dz \\ &= \int_0^S T(\mathbf{x_z}) L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega})\, dz + \int_0^S T(\mathbf{x_z}) L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega})\, dz \\ &= L_{MS}(\mathbf{x_S}, \boldsymbol{\omega}) + L_{SS}(\mathbf{x_S}, \boldsymbol{\omega}). \end{aligned} \tag{3.2}$$

In conclusion, this combination results in a physically-based yet efficient technique for the real-time synthesis of the two main components of underwater appearance.

For the full multiple scattering radiance $L_{MS}(\mathbf{x_S}, \boldsymbol{\omega})$, we propose a physically based analytical approximation based on oceanography (see next chapter, 4 for more details). This approximation is derived in a formal way, allowing a highly controllable real-time spectral simulation of underwater light transport (each wavelength computation becomes O(1)). We validate it against a spectral Path Tracer in Mitsuba2 to show the importance of spectral rendering in underwater scenarios.

Single scattering $L_{MS}(\mathbf{x_S}, \boldsymbol{\omega})$, on the other hand, varying both spatially (due to the deformations in the water surface) and temporally (due to the movement of the waves) cannot be properly modeled analytically. Therefore, we leverage state-of-the-art real-time single scattering techniques to provide a plausible approximation. In this case, we leave out the spectral aspect, as it would be too expensive to work in real time. We will detail this method in chapter 5.

4. Multiple scattering approximation

In this work, we propose to approximate multiple scattering using the physically measurable vertical diffuse attenuation coefficient of downwelling irradiance $K_d(\lambda)$. This coefficient measures the irradiance (radiance from every direction) attenuated in the water volume per vertical meter for wavelength λ. It is usually derived from the combination of backscattering and absorption coefficients [Mob94; LDC*05].

In oceanography, optical properties are often classified in Inherent Optical Properties (IOPs) and Apparent Optical Properties (AOPs). Inherent Optical Properties are invariant to outside factors. For example, the scattering coefficient of a medium does not depend on the sun radiance or zenith angle. AOPs, on the other hand, may vary depending on the outside radiance distribution, although they must be robust enough that they are useful descriptors of the medium [Mob01].

The diffuse downwelling attenuation coefficient is an Apparent Optical Property. However, it is often considered a quasi-Inherent Optical Property due to its robustness [Gor89]. Therefore, it is an especially appropriate measured coefficient to model underwater illumination in different media.

If we note $E_D(\lambda, y)$ the irradiance of wavelength λ at depth y, we have

$$E_D(\lambda, y) = E_{D_0}(\lambda) e^{-K_d(\lambda) y}, \tag{4.1}$$

where $E_{D_0}(\lambda)$ is the incident downwelling irradiance just below the surface of the water body [Blo19]. This irradiance is defined for a horizontal plane at depth y with its normal pointing towards the water surface. In our case, we ignore this directionality, as we are considering that diffuse illumination underwater is constant around the sphere. This leads to the following approximation:

$$\int_{\Omega} L(\mathbf{p_y}, \boldsymbol{\omega}_i)\, d\boldsymbol{\omega}_i \approx E(y). \tag{4.2}$$

where $\int_{\Omega} L(\mathbf{p_y}, \boldsymbol{\omega}_i)\, d\boldsymbol{\omega}_i$ is the incident radiance from every direction of the unit sphere Ω. This is illustrated in Figure 4.1.

Now, if we recall Equation (3.1), we are trying to find the total multiple scattering radiance L_{MS},

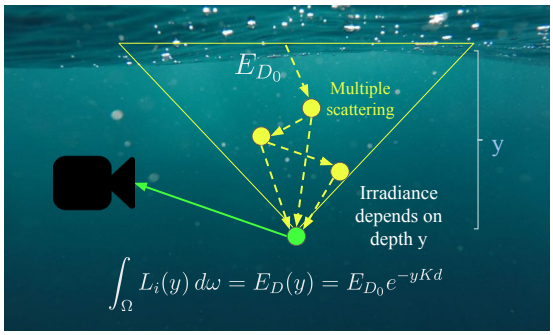


Figure 4.1: We approximate ambient illumination at depth y using the diffuse downwelling attenuation coefficient kd from oceanography.

$$L_{MS}(\mathbf{x_S}, \boldsymbol{\omega}) = \int_0^S T(\mathbf{x_z}) L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega})\, dz. \tag{4.3}$$

We can expand the in-scattering radiance at each point $L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega})$ using the isotropic phase function and our irradiance approximation from Equation (4.2):

$$\begin{aligned} L_{iMS}(\mathbf{x_z}, \boldsymbol{\omega}) &= \int_\Omega f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}) L(\mathbf{p_y}, \boldsymbol{\omega}_i)\, d\boldsymbol{\omega}_i \\ &= \frac{\sigma_s}{4\pi} \int_\Omega L(\mathbf{p_y}, \boldsymbol{\omega}_i)\, d\boldsymbol{\omega}_i \\ &= \frac{\sigma_s}{4\pi} E(y) \\ &= \frac{\sigma_s}{4\pi} E_{D_0} e^{-K_d(\lambda) y} \end{aligned} \tag{4.4}$$

Notice that we need y, the distance of point $\mathbf{x_z}$ to the water surface. To obtain it, it is useful to express the point as a function of the origin $\mathbf{o}$ of the camera instead. In this case, from the equation of the ray, $\mathbf{x_z} = \mathbf{o} - z\boldsymbol{\omega}$ (as vector $\boldsymbol{\omega}$ is pointing towards the camera). Therefore, the distance to the surface will be $y = o_y - z\omega_y$, where o_y and ω_y are the vertical coordinates of the origin and the direction vector, respectively. We can now rewrite Equation (4.4) as:

$$L_{iMS}(\mathbf{o} + z\boldsymbol{\omega}, \boldsymbol{\omega}) = \frac{\sigma_s}{4\pi} E_{D_0} e^{-K_d(\mathbf{o}_y - z\omega_y)} \tag{4.5}$$

With this new parametrization, substituting into Equation (4.3), and including the transmittance term as defined in Equation (2.10), we obtain the following expression:

$$\begin{aligned} L_{MS}(\mathbf{o} + S\boldsymbol{\omega}, \boldsymbol{\omega}) &= \int_0^S T(\mathbf{x_z}) L_{iMS}(\mathbf{o} + z\boldsymbol{\omega}, \boldsymbol{\omega})\, dz \\ &= \int_0^S e^{-\sigma_t z} \frac{\sigma_s}{4\pi} E_{D_0} e^{-K_d(\mathbf{o}_y - z\omega_y)}\, dz. \end{aligned} \tag{4.6}$$

4.1 Analytical solution

Finally, by regrouping every term independent from z, we can analytically solve the resulting exponential integral:

$$\begin{aligned} L_{MS}(\mathbf{o}+S\boldsymbol{\omega},\boldsymbol{\omega}) &= \int_0^S e^{-\sigma_t z}\frac{\sigma_s}{4\pi}E_{D_0}e^{-K_d(o_y-z\omega_y)}\,dz \\ &= \frac{\sigma_s}{4\pi}E_{D_0}\int_0^S e^{-\sigma_t z}e^{-K_d(o_y-z\omega_y)}\,dz \\ &= \frac{\sigma_s}{4\pi}E_{D_0}\int_0^S e^{-K_d o_y+(K_d\omega_y-\sigma_t)z}\,dz \\ &= \frac{\sigma_s}{4\pi}E_{D_0}e^{-K_d o_y}\int_0^S e^{(K_d\omega_y-\sigma_t)z}\,dz \\ &= \frac{\sigma_s e^{-K_d o_y}E_{D_0}}{4\pi(K_d\omega_y-\sigma_t)}(e^{(K_d\omega_y-\sigma_t)S}-1) \end{aligned} \tag{4.7}$$

As we can see, this approximation has allowed us to reduce the two integrals in Equation (2.11), to an efficient O(1) (per wavelength) expression. This computes the full contribution from the medium for any given ray given the σ_s, σ_t and K_d coefficients that define the optical properties of the medium; and the position $\mathbf{o}$ and viewing direction $\boldsymbol{\omega}$ of the sensor, as well as and the length of the ray S, which together encode the depth under the water surface.

This is the key expression that, while remaining physically-based, can be evaluated in constant time and therefore enable our real-time spectral rendering.

4.2 Spectral to RGB conversion

Until now, we have omitted the wavelength dependency from the equations for simplicity. In reality, light is composed of a spectrum of varying wavelength: all medium coefficients (σ_a, σ_s, K_d), and all radiance terms in the equations (L_m, L_s, etc) are wavelength-dependent. Figure 4.2 (left) shows a plot of a medium's coefficients.

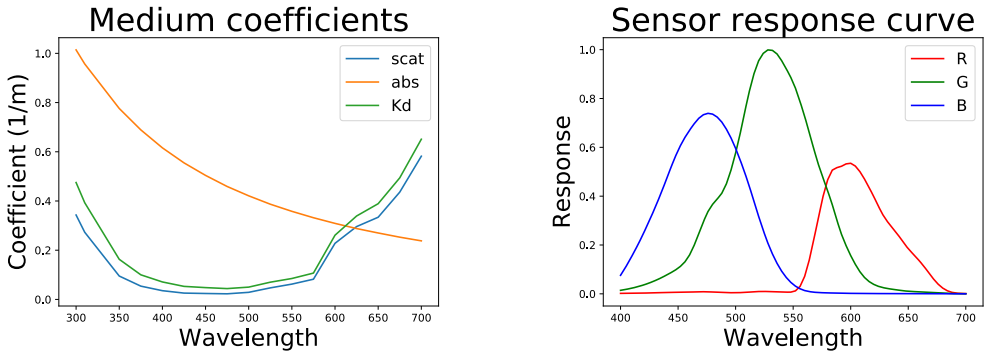


Figure 4.2: Example spectral data. Left: Medium spectral coefficients. Right: RGB response curve of a camera.

It is common to work in RGB, as it fits our perception and is fast to compute (especially in real time applications). However, integrating a higher number of spectral samples results in a more accurate approximation of the underlying spectrum.

More specifically, we can divide the spectrum in n wavelength bands and evaluate Equation (4.7) once for each of them, with the coefficients of that wavelength band. This results in n values for every pixel instead of the typical 3. In order to display it, we need to bring it back into RGB, which is achieved by applying the spectral sensitivity function of the sensor.

Given a sensor response $f(\lambda)$ (Figure 4.2) that maps an intensity of wavelength λ to its RGB value, and a rendered spectrum $L(\lambda)$, the final RGB values are mathematically the convolution of both functions over the wavelength domain:

$$L_{RGB} = \int_{\lambda} L(\lambda) f(\lambda)\, d\lambda \approx w \sum_{\lambda} L(\lambda) f(\lambda). \tag{4.8}$$

We approximate this integral using the rectangle rule, which estimates the resulting area by discretizing the resulting function into n bands of width $w = \frac{max(\lambda) - min(\lambda)}{n}$ and adding the area of their corresponding rectangles.

This is a typical integration method that can be used for arbitrary functions. In Figure 4.3, we can see two visual examples of the method, applied to a function plotted in blue. Finding the integral of the function is, by definition, finding the area under the curve, which is approximated by the n rectangles. On the left, we use $n = 4$ and, on the right, $n = 40$, yielding a much more accurate approximation.

In Equation (4.8), we use the exact same method for each of the three RGB channels.

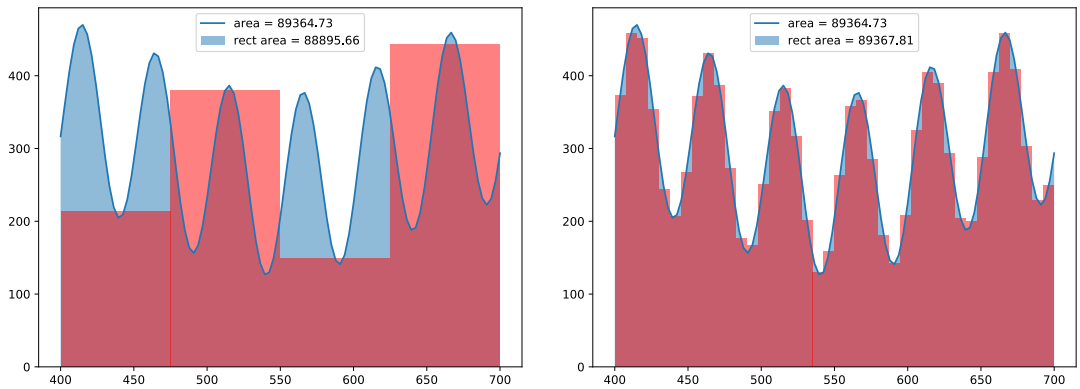


Figure 4.3: Two examples of applying the rectangle rule to a given function (plotted in blue). Left: using only four samples, the estimation is less accurate. Right: Using 40 samples, the area of the rectangles is much closer to the actual function. Using infinitely many rectangles would yield the exact area under the function.

4.3 Multiple scattering implementation overview

4.3.1 Engine selection

Our proposed method could be implemented in any real time rendering engine. While a low level library, such as OpenGL or Vulkan, would offer more control over every detail of the implementation, a more general purpose engine such as Unity offers a lot of features, allowing us to focus our efforts on implementing the rendering code.

These engines feature rendering pipelines, which are responsible for drawing the current scene to the screen every frame, responding to the lighting, scene geometry, materials and so on. The pipelines are usually divided in sequential operations on the data, which are called rendering passes (e.g., an anti-aliasing pass reduces the aliasing of the final image). Often, the intermediate data along the pipeline are textures which store relevant data for every pixel. For example, the depth buffer contains the distance to the camera for each pixel.

Unity has several rendering pipelines. The Built-in rendering pipeline is older and generalist, but it does not allow for much customization, only writing some specific shaders (for example for materials or visual effects). Unity has since introduced the Scriptable Rendering Pipelines, which are controllable from scripts and thus offer much more customization options. There are two scriptable rendering pipelines by default: URP (Universal), and HDRP (High Definition).

URP focuses on performance and is compatible with lower end devices such as smartphones, while HDRP focuses on physically based rendering for higher end systems. We have opted for HDRP, as our project is also focused on physically based rendering.

HDRP makes use of compute shaders, which are a high level abstraction of the compute kernels able to receive and output arbitrary data (instead of being specifically tailored for rendering, as typical shaders are). This also fits our problem, as we need to input our hyperspectral medium coefficients and camera response curves into the shader.

4.3.2 Our Underwater Rendering Pipeline (UWRP)

Deferred rendering Our pipeline, the Underwater Rendering Pipeline (UWRP) is therefore a modified version of HDRP, tailored for our needs. One modification is the type of pipeline. Rendering pipelines can be forward or deferred depending on the organization of the render passes.

Forward pipelines are the historically default real-time implementation, and they apply the different passes sequentially to each object to be rendered. A high level diagram is shown in Figure 4.4: each of the objects is processed individually, which makes the fragment shader of each of the objects process every light (i.e., the cost is $O(\#lights\#objects)$).

Deferred pipelines, on the other hand, first gather all the *pixel-wise* data involved in the lighting (normals, depth, albedo...), store it in the geometry buffer (G Buffer), and *defer* the lighting computations to the deferred shader. This pipeline is shown in Figure 4.5. As we can see, the use of the GBuffer increases the memory footprint of the solution, but has the main

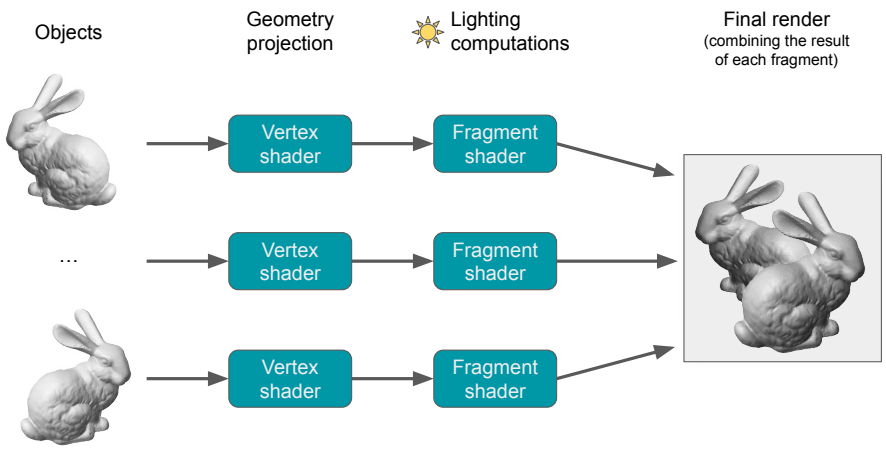


Figure 4.4: Forward rendering computes the lighting of each object (bunnies on the left) separately. First, the vertex shader projects the geometry into the camera. Then, the fragment shader performs the lighting computations for each of the projected pixels. The resulting colors are mixed into the final render. Notably, this means that the full lighting computations are processed by the fragment shader of each of the objects.

advantage of only performing the lighting computations in a single shader (making the pipeline $O(\#objects + \#lights)$).

HDRP supports both types of pipelines, as both of them have their use cases. For example, translucency is much easier to implement in a forward way: if we have a scene with a translucent window and a car behind, the forward pipeline would draw both independently, and then blend them together. This can even be done recursively for more translucent objects. However, this is a harder problem for deferred shading, as all the lighting information cannot be easily stored in the G Buffer anymore (it depends on several different surfaces which can not be known *a priori*).

In our case, deferred rendering is more appropriate, as it ensures that our spectral computations are only computed once per pixel. Additionally, effects like translucency are out of the scope of this project.

Pipeline overview In Figure 4.6, we show a high level overview of our UWRP for multiple scattering. First, the intermediate data is collected and processed in a series of textures (albedo, depth map), arrays (medium coefficients and sensor response curve) and other data (camera matrix, number of wavelengths n, etc.).

This data is sent into our deferred shader, which evaluates Equation (4.7) for each of the n wavelengths. This is convolved with the response curve as described in Equation (4.8) (section 4.2), resulting in the RGB render.

Main modifications In more detail, we have made the following modifications to HDRP. UWRP now contains an OceanClass, which encloses all the ocean related data and functionality on the CPU side. It also acts as an interface to allow application code to interact with it (keeping

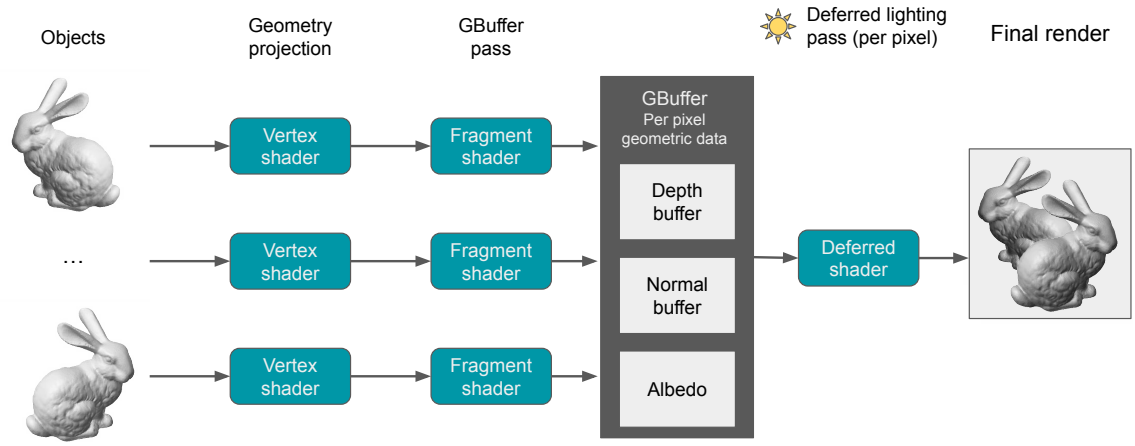


Figure 4.5: Deferred rendering instead renders a number of intermediate buffers with geometric information (normals, depth, albedo...). These values are combined into the GBuffer. This buffer effectively encodes the whole scene in screen space. The deferred shader uses this to perform the lighting computations per pixel, independent of the number of objects.

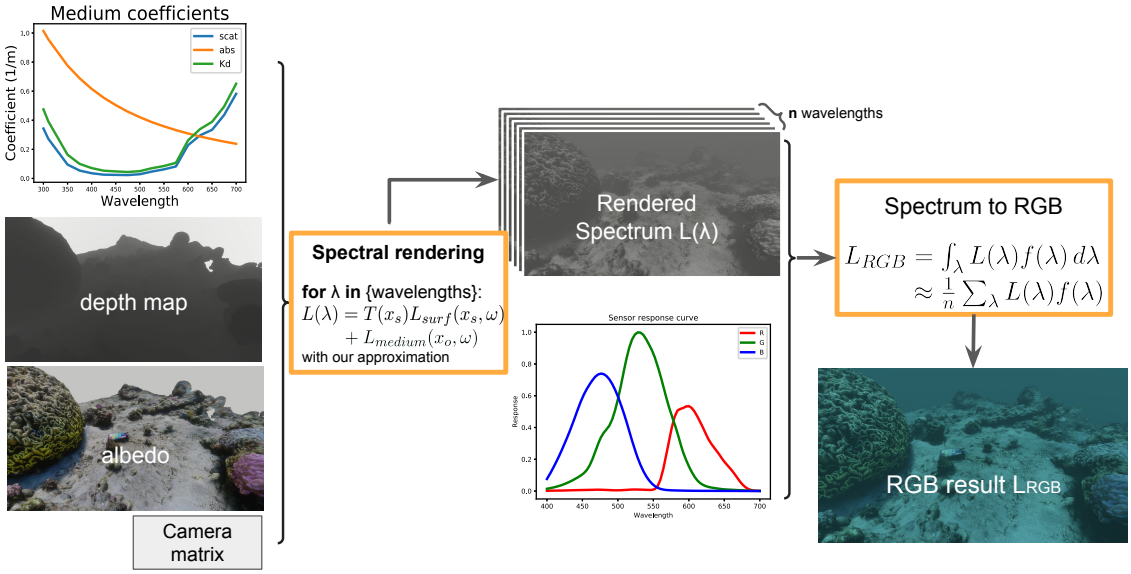


Figure 4.6: Overview of our UWRP.

a separation from the engine code), which for example enables its interaction through a user interface.

It itself contains the SpectralData class, which handles loading the coefficients and response curves from disk, and normalizes them to the same wavelength range, handles interpolation of the wavelength bands depending on n.

On the GPU side, we have added the new data to the deferred shader. It only receives the curated data, as shown in Figure 4.6.

Advantages of modifying HDRP It may seem overcomplicated to modify Unity's HDRP instead of coding our UWRP from scratch. However, this implementation enables our pipeline to take advantage of most of HDRP's features.

A simple example is tone mapping. Lighting computations are done in High Dinamic Range (HDR): unbounded floating-point luminance values. However, they need to be remapped to the $[0, 1]$ range in order to be displayed in the screen. Tone mapping is the operation that remaps any color value to its appropriate range, and has been an active field of study both in computer graphics and in photography [RSSF02]. In HDRP, tone mapping is provided as a postprocessing effect (a render pass applied to the rendered frames just before drawing them to the screen).

Furthermore, this enables us to use HDRP's volumetric lighting architecture. In the following chapter, 5, we will detail how we leverage it to obtain the single scattering radiance L_{SS} to combine it with the previous multiple scattering radiance L_{MS}, completing our proposed pipeline (Chapter 3).

5. Single scattering approximation

With our multiple scattering approximation, we are considering *diffuse* downwelling irradiance, which accounts for the lighting reaching each depth coming from the diffuse radiance just below the surface. However, especially near the surface, there is a lot of high frequency spatial variation due to the volumetric caustics and shadows caused by the refractions in the deforming surface. See Figure 5.1 for a reference image of the effect we are emulating.

Figure 5.1: Underwater photograph featuring the single scattering caustics we are emulating (often called god rays or lightshafts). Photograph by Chris Kippax in Wetpixel.

These effects are however very costly to simulate, as they vary both spatially and temporally, so some physical accuracy has to be sacrificed in favor of performance.

More formally, if we recall the medium VRE from Equation (3.2), we are approximating the following integral for the single scattering radiance:

$$L_{SS}(\mathbf{x_S}, \boldsymbol{\omega}) = \int_0^S T(\mathbf{x_z}) L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega})\, dz. \tag{5.1}$$

In the multiple scattering case, by considering in-scattering as a function of only depth, and ignoring occlusions, we were able to find an analytical solution to the resulting integral (4.1). In this case, we need to account for these occlusions and more complex spatial variations in the incoming light field (due to the moving caustics).

To illustrate the problem, consider the simple example of Figure 5.2, where we have the sun as a light source, the camera, three occluders (in black), and a homogeneous participating medium everywhere. Due to these occlusions point $\mathbf{x_a}$ is illuminated by the sun, but points $\mathbf{x_b}$ and $\mathbf{x_c}$ will not. In terms of Equation (5.1), the problem is that $L_{SS}(\boldsymbol{\omega}_1)$ (in orange) and $L_{SS}(\boldsymbol{\omega}_2)$ (in blue) depend on the in-scattering $L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega})$ at every differential point $\mathbf{x_z}$ along their rays. In this case, we can see that $L_{SS}(\boldsymbol{\omega}_1)$ will receive radiance because it accounts for the in.scattering of some unoccluded points, such as $L_{iSS}(\mathbf{x_a}, \boldsymbol{\omega})$. On the other hand, $L_{SS}(\boldsymbol{\omega}_2) = 0$, as every point on its path is occluded (as $L_{iSS}(\mathbf{x_c}, \boldsymbol{\omega}) = 0$).

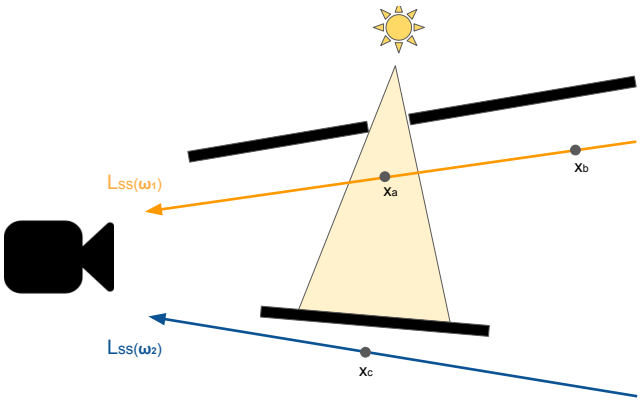


Figure 5.2: Illustrative example of the problems posed by considering spatially-varying single scattering. For this simple scene with the camera, three occluders (black), and the sun, points $\mathbf{x_b}$ and $\mathbf{x_c}$ will be occluded, whereas $\mathbf{x_a}$ will receive radiance: $L_i(\mathbf{x_a}) > 0$. Therefore, $L_{ss}(\boldsymbol{\omega}_1)$, in orange, on top, will receive radiance accounting for x_a; $L_{ss}(\boldsymbol{\omega}_2)$ will not, as every point along its ray is occluded. The issue is that *every* differential point besides the three listed will need to be accounted for.

Therefore, the in-scattering terms $L_{iSS}(\mathbf{x_a}, \boldsymbol{\omega}_i)$ need to somehow account for the visibility to the light sources (as well as consider the intensity and extinction towards those light sources). In practice, these occlusions cannot be modeled analytically, as they depend on every lightsource on the scene and the geometry of the occluders. Therefore, we need some numerical approximation.

5.1 Froxel lighting

We utilize a standard solution to this problem, which is widely utilized in game engines, called froxel lighting [Vos14; Hil15; Kov20], and is implemented in HDRP. Froxel lighting is a two pass algorithm which first computes a discrete representation of the in-scattering L_{iSS} in the scene, and then traverses this representation to approximate the final single scattering integral L_{SS}.

The discretization of the space can be illustrated following the previous example. As we can see in Figure 5.3, space is subdivided in froxels: voxels in frustum space. Voxels are the typical cubic subdivision of 3D space, but in this case they are applied to the deformed space that is visible to the camera, the camera frustum. This subdivision, instead of using world space voxels, has the great advantage of naturally bringing a higher spatial resolution where it is most needed, near the camera. In the figure, only the three center froxels will receive lighting for the sun, and therefore, only points such as $\mathbf{x_a}$ will account from in-scattering from the sun.

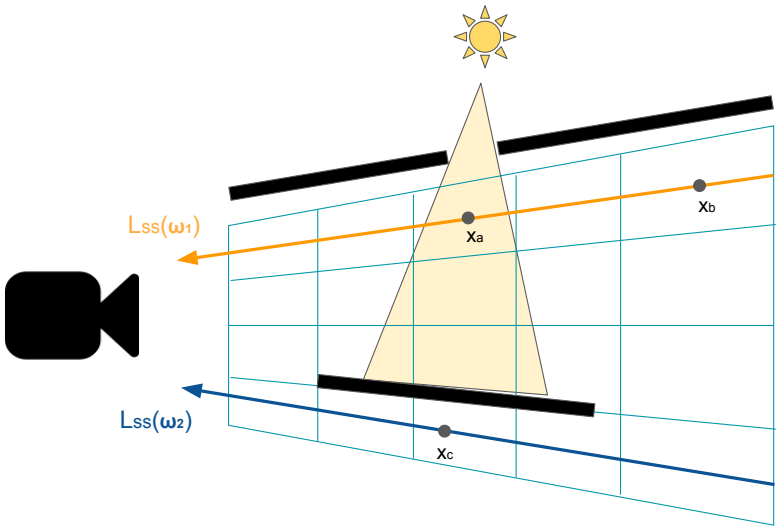


Figure 5.3: Illustration of the froxel technique which allows for volumetric, spatially varying, single scattering. For a given camera, its frustum is divided in froxels (blue grid in the illustration). The incoming radiance of each froxel is computed in the first part of the algorithm. For example, in the illustration, one of the froxels will contain the radiance of $L_{iSS}(\mathbf{x_a})$, that is unoccluded. In the integration pass, the froxels will be traversed to integrate their illumination (e.g., the ray in the bottom will receive no radiance, as all of its froxels will be occluded $L_{SS}(\boldsymbol{\omega}_2) = 0$, while radiance $L_{SS}(\boldsymbol{\omega}_1)$, in orange, will account from the sun in the center column).

The second step of the froxel lighting algorithm approximates Equation (5.1) by traversing these discrete froxels (mathematically, this is again an application of the rectangle rule, see section 4.2):

$$\begin{aligned} L_{SS}(\mathbf{x_S}, \boldsymbol{\omega}) &= \int_0^S T(\mathbf{x_z}) L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega})\, dz \\ &= d \sum_{\mathbf{x_z} \in XV(\boldsymbol{\omega})} T(\mathbf{x_z}) L_{iSS}(\mathbf{x_z}, \boldsymbol{\omega}) \end{aligned} \tag{5.2}$$

where $XV(\boldsymbol{\omega})$ are the set of froxels for direction $\boldsymbol{\omega}$, and d the length of each froxel along the camera z axis.

This technique has its limitations. Mainly, it is an extremely expensive process in a high resolution grid: the order is $O(\#voxels \#lights)$ for lighting the voxels and an additional $O(\#voxels^{\frac{1}{3}})$ to integrate each ray, where $\#voxels$ is the number of voxels per side of the

cube. In practice, a very low resolution grid has to be used, and some techniques exist to improve its quality.

5.2 Single scattering implementation overview

A version of froxel lighting is implemented in HDRP [Lag18]. They do an additional previous step to the volumetric lighting, which processes the different volumes present in the scene, and returns their density. Then, the froxel lighting step of the algorithm is a rendering pass that returns the in-scattering L_{iSS} of each froxel in the VBuffer, a 3D texture which is the the volumetric equivalent of the GBuffer.

In our case, as we have modified our UWRP (section 4.3.2) to compute our spectral rendering approximation, we need to compute $L_{SS}(\mathbf{x}, \boldsymbol{\omega})$ according to the in-scattering data encoded in this VBuffer. To achieve this, we have added the VBuffer data to our shader and evaluate Equation (5.2) in our deferred shader.

Finally, this results in high quality single scattering $L_{SS}(\mathbf{x}, \boldsymbol{\omega})$ that reacts to all lights and occluders in the scene. A limitation of HDRP's method is that it does not account for the extinction between the light source and the froxel being lit. In our case, this would result in volumetric caustics independent on depth underwater.

However, as we use this method to render caustics caused by the sun, we know the distance and angle to the light source, and we can compensate it when computing L_{ss}.

HDRP's volumetric framework defines participating media through their single scattering albedo a and mean free path (average distance depending on the extinction coefficient: $MFP = 1/\sigma_t$). Therefore, the only missing step is transforming our spectral coefficients into this data:

$$a = \int_\lambda \frac{\sigma_s(\lambda)}{\sigma_t(\lambda)} f(\lambda)\, d\lambda \tag{5.3}$$

$$MFP = \frac{1}{\int_\lambda \sigma_t(\lambda) f(\lambda)\, d\lambda} \tag{5.4}$$

recall that $f(\lambda)$ is the response curve of the camera. With this data, we create an HDRP Volumetric Fog object, which enables us to obtain the water's single scattering lighting following the overview in Figure 5.4. Recall that section 5.1 explains the details of the technique.

Integrating this with our multiple scattering lighting into our Underwater Rendering Pipeline enables us to realistically simulate the single scattering lighting of any light source, as we can see in Figure 5.5.

5.3 Caustics

Many works in the literature focus on efficiently rendering caustics [IDN02; IDN03; Wat90; EAJ05; GLF*08]. In real time, the most common technique (for its simplicity) is projecting a

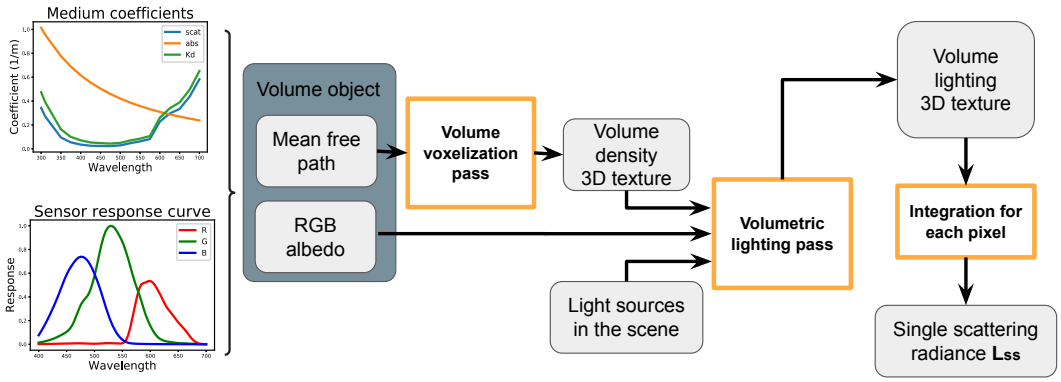


Figure 5.4: High level overview of the pipeline to generate our single scattering radiance L_{ss}. Data (textures or values) are grey, and render passes have an orange outline. First, the spectral coefficients of the medium and the response curve are used to compute the albedo and mean free path of the medium, to generate an equivalent HDRP volume object (Fog). The volume voxelization pass generates the appropriate froxels from the density data. Then, the scene lighting and the froxels are combined in the volumetric lighting pass. Finally, each pixel in the scene integrates the intersected voxels to generate the final single scattering lighting.

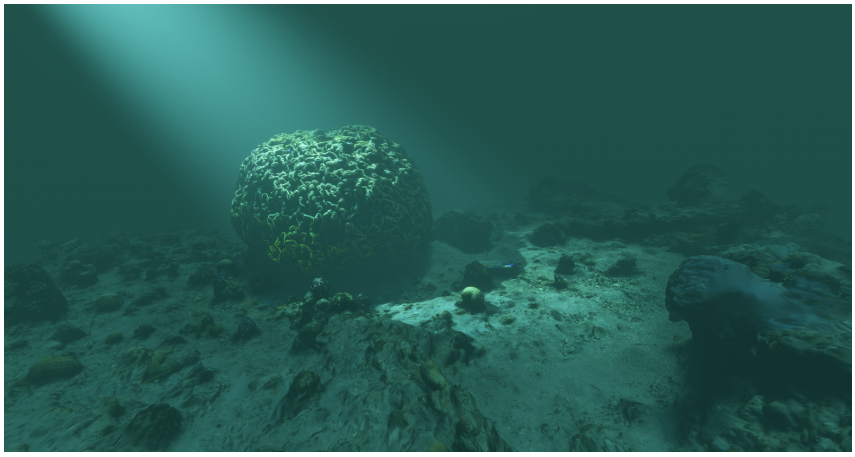


Figure 5.5: Render of a scene with a spotlight with single scattering.

moving texture into the surfaces, introduced by Stam in 96 [Sta96].

In our case, we can use HDRP's light cookies (textures to spatially modulate the emission of lights) and focus on generating the pattern of radiance trespassing the ocean surface. Our pipeline will properly react to the radiance on both solid surfaces and the water medium, as well as handling shadows.

Our light cookie is attached to the sun, and is itself a render texture. Render textures are a procedural textures updated every frame by a shader. In the shader, we use a standard technique to generate procedural animated caustic patterns [Ame22]. Two instances of a seamless caustic texture are offset and move in different directions and at different speeds. The final value at each pixel, which will modulate the intensity of the sun at a given point, is computed as the minimum of both textures.

This results in a believable pattern of directional caustics, as we can see in Figure 5.6. As we account from the extinction from the surface, the greater depths are multiple-scattering-dominated.

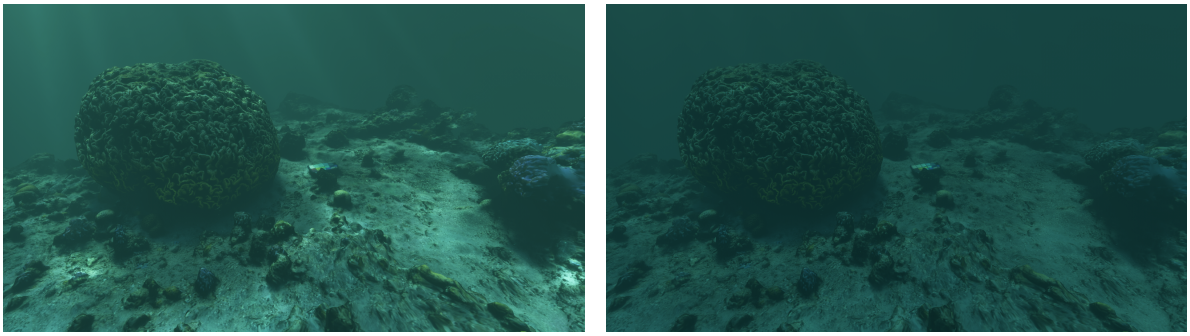


Figure 5.6: Render of a scene with our caustics method. Left: 5m depth. Right: 12 m. As we can see, single scattering is more prevalent near the surface.

6. Results

Our method achieves high quality results at interactive frame rates (around 50 frames per second without single scattering and around 30 with it). Our physically based pipeline is able to simulate the appearance of the varied Jerlov water types [SM15]. There are ten water types, five of them from open oceans (I, IA, IB, II, III), and five coastal (1C, 3C, 5C, 7C and 9C), all of them ordered from clearest to most turbid. I, IA and IB are absorption dominated, the rest, scattering dominated. In Figure 6.1, we show example renders of the open waters (one in each column) at different depths (one at each row). The same sensor has been used. All renders can be found in full resolution in the additional materials.

In Figure 6.2, we show the equivalent table for the coastal waters. As we can see, the pipeline can simulate a great variety of water media and depth. The single scattering component rapidly becomes less prevalent in lower depths.

In Figure 6.3, we show two example scenes, one scattering dominated (left) and the other absorption dominated (right). Both are using the same response curve. For reference, we show the same scenes rendered using a spectral Path Tracer (top), which does not make use of our approximation, but numerically solves the RTE by Monte Carlo integration. Our method is the second row. As we can see, we obtain perceptually similar results under the same physical conditions.

It should be noted that the path traced scenes took around 4 hours to simulate, whereas our real time method generates the frames in 0.03 s. This represents a speedup of the order of 10^6, although this comparison is of course not entirely fair, as the path tracer runs on the CPU.

In the third row, we show the result of implementing our method in RGB. It works well for the most turbid water, but it shows clear differences with respect to the reference on the more clear Jerlov IB scene.

In the additional materials (https://nas-graphics.unizar.es/s/nestor-TFM), we show all the rendered screenshots from our method, the RGB implementation, and the Path Tracer on a larger set of media and camera response curves.

6.1 Quantitative evaluation

For a more quantitative evaluation, in this section, we compare our multiple scattering spectral approximation with the RGB implementation using the Path Tracer simulation as ground truth.

Figure 6.1: Renders of the different Jerlov open waters (I to III), one in each column. The rows show increasing depth.

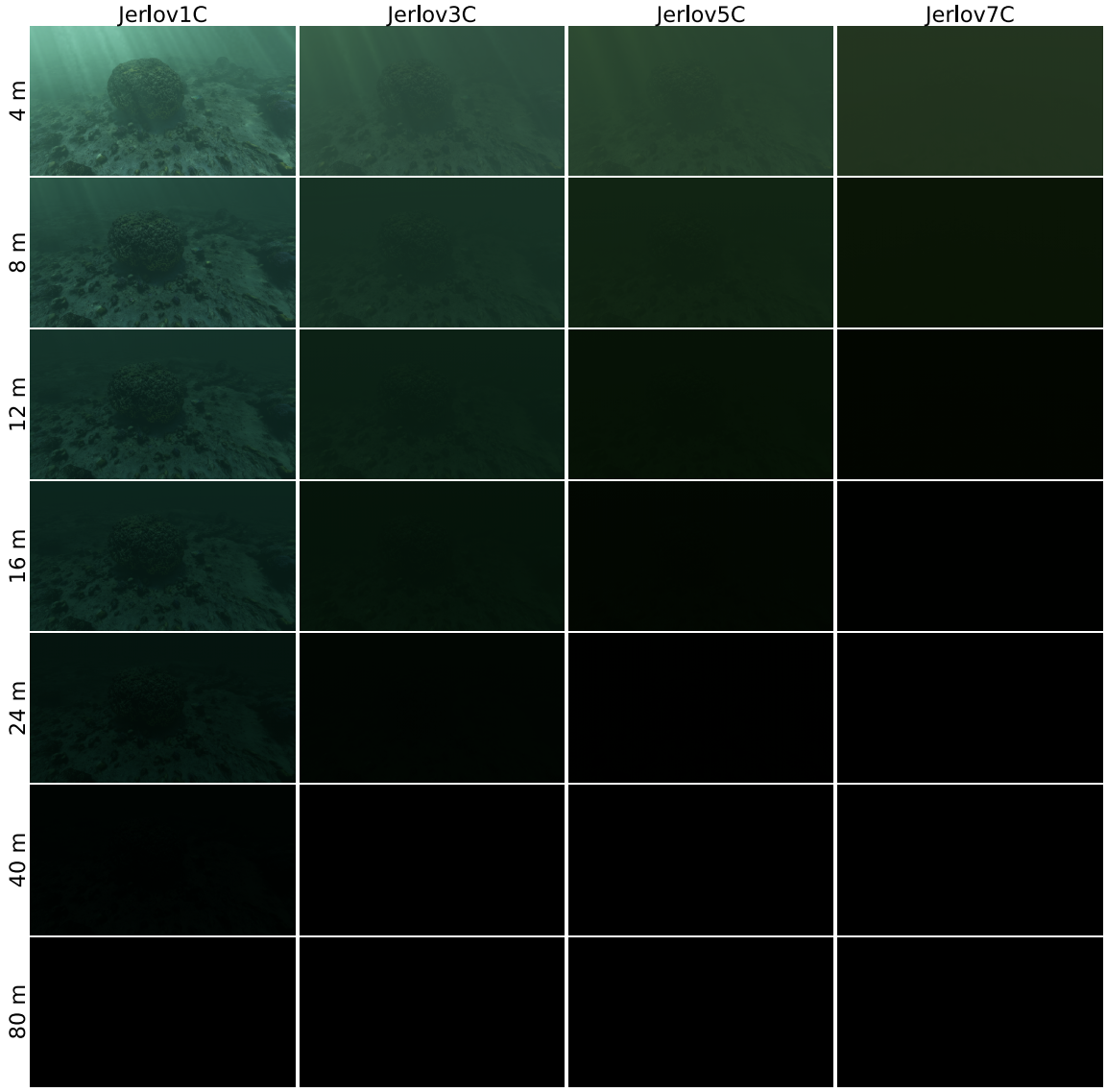
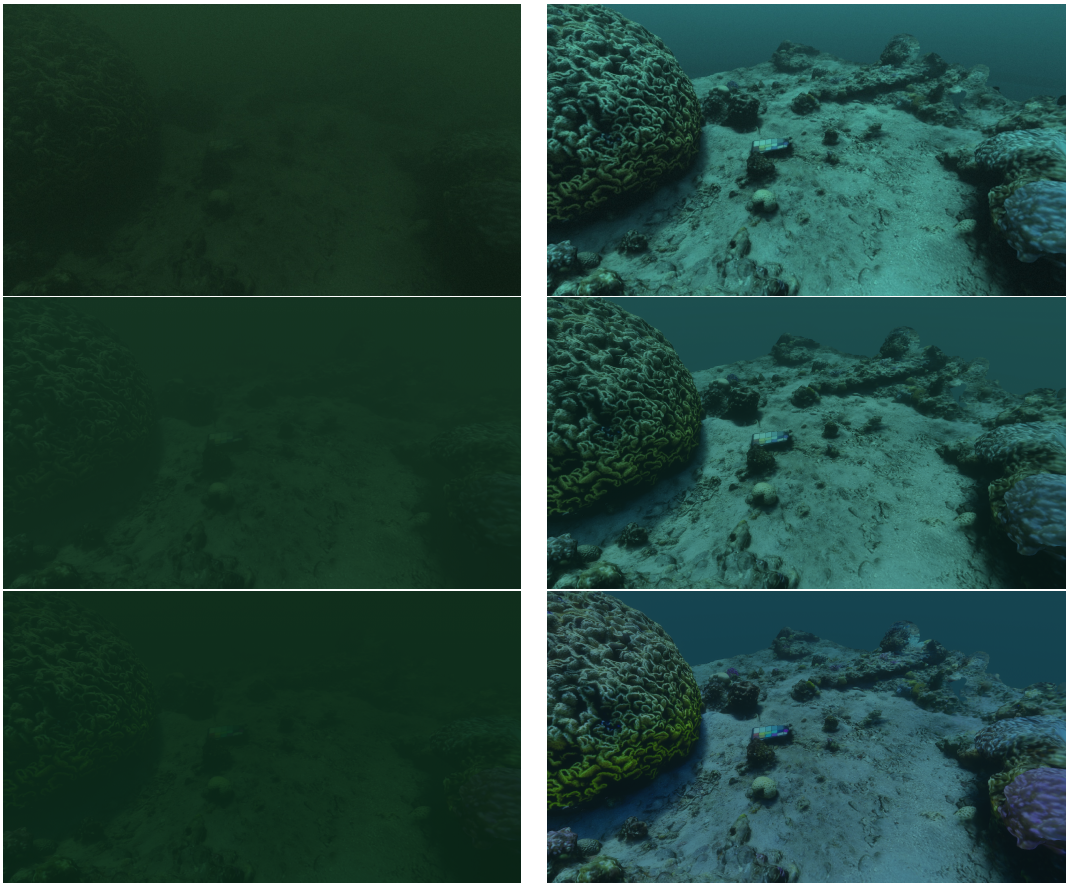


Figure 6.2: Renders of the different Jerlov coastal waters (1C to 7C, 9C omitted due to its huge turbidity), one in each column. The rows show increasing depth.



(a) Jerlov 3C, depth=9m

(b) Jerlov IB, depth=5m

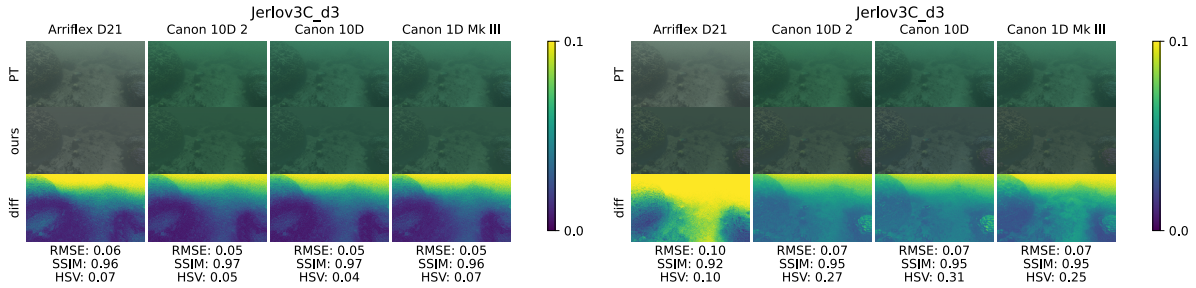
Figure 6.3: Example renders of a scattering dominated medium (left) and an absorption dominated one (right). Top: reference path traced render ($\sim$ 4 h). Second row: ours ($\sim$ 0.02 s). Third row: real-time RGB implementation ($\sim$ 0.02 s).

We leave out the single scattering method in this case, as our reference does not have an implementation of this effect. We study six combinations of medium coefficients and underwater depths (3, 5 and 9 meters).

The media studied include two scattering-dominated waters (Jerlov 3C and Jerlov II), and three absorption-dominated waters (I, IA, IB). Given a scene (medium and depth), it is rendered using the response curves of 28 different cameras, and each of those is compared and summarized.

Given a real-time render and the path traced reference, first we take the pixel-wise difference between the two. Then, we extract two metrics: the RMSE (root mean squared error) and an HSV distance. For the HSV metric, we convert both colors to the HSV color space and compute the distance of the Hues (taking into account it is a cyclic value) and add the Saturation distance. We ignore the Value, as we focus on the color change and not its brightness, which could be compensated by exposure. More precisely, we map the HSV colors to carthesian coordinates first, normalizing the Value channel, and then compute the euclidean distance there, in the resulting circle.

In Figure 6.4, we can see some of the cameras being compared for one of the scenes (Jerlov3C, $depth = 3m$), as well as these metrics. SSIM, the Structural Similarity Index Measure has also been computed.



(a) Spectral

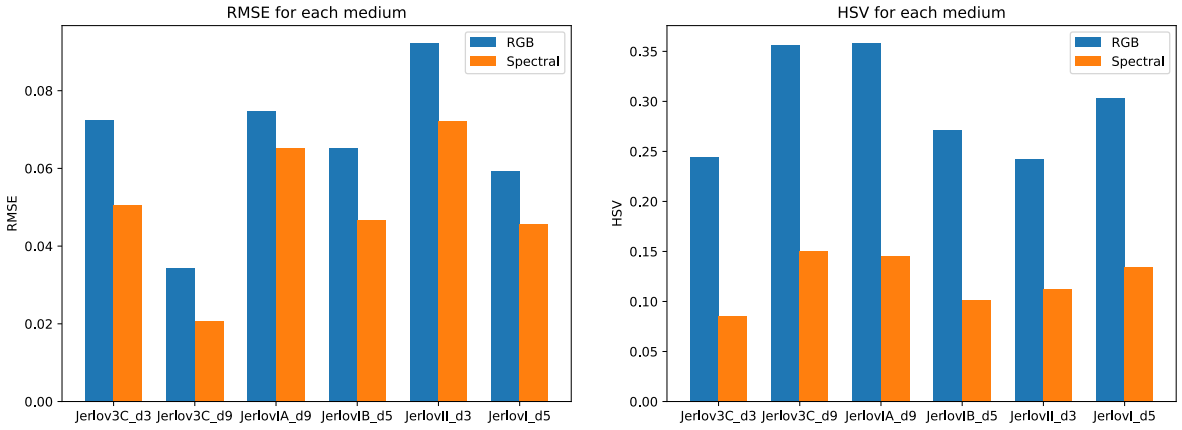
(b) RGB.

Figure 6.4: Example of the computed metrics for several cameras on the Jerlov3C, $depth = 3m$ scene. The rows are: Path traced, real time, difference. The difference images are a heatmap of the absolute error of each pixel (in the RGB color space). The real time row is our method on the left, and the RGB implementation on the right.

The SSIM is a perceptual metric which measures how similar two images look, ranging from -1 (not similar) to 1 (very similar). It is a robust metric, as it does not require the pixels to be perfectly aligned as the other metrics do. However, in this case, we have found it to not be very informative, as differences in the color are less important for this measure.

In Figure 6.5, we show a summary of the RMSE and HSV based metrics, computed for all of the cameras for each of the six tested scenes (each abbreviated as before, where the Jerlov3C scene, at $depth = 9$ meters becomes *Jerlov3C_d9*). In the plots, the RGB implementation is in blue, while our spectral method is in orange. As we can see, our method achieves a significantly higher accuracy with both metrics, in all of the tested scenes. The HSV error difference is the most noticeable, the spectral version reduces it to approximately half of the one of the RGB implementation.

This suggests that spectral rendering improves accuracy in underwater rendering.



(a) RMSE

(b) HSV comparison.

Figure 6.5: Summary of the average RMSE and the hue and saturation difference (in the HSV space) of both methods, for each of the tested media. The RGB implementation is in blue, and our spectral method in orange.

6.2 Comparison with USSim

USSim [CZSZ22] (which we introduced briefly in the Related Work section 1.1), is a method which is similar in spirit to ours:

- They tackle the same problem: underwater visualization in real time.
- They also propose a physically based imaging model which is based in the diffuse downwelling attenuation coefficient for ambient lighting.
- It is also implemented in Unity (albeit not in HDRP).

However, their work is limited for two main reasons. First, they do all their computations using the three RGB wavelength bands, which limits its hyperspectral applications, and may introduce errors in absorption dominated waters. Secondly, they compute a single ambient illumination color A based on their *depth* parameter, which is modifiable but *constant* for the whole scene. On the other hand, we have properly parameterized our model so that it accounts for the real distance to the surface from any point in the scene. This makes it especially interesting for dynamic scenes of varying depth, as well as more physically accurate.

To compare our methods, we have used their demo scene with both their pipeline and ours. In Figure 6.6, we can see a comparison using three different water media, and two depths for each, resulting in the six columns. The chosen Jerlov water types are: 1C (coastal, low turbidity), IA (open, clear) and II (open, more turbid). In the first row we show our method and, in the second, USSim.

We will only perform a qualitative analysis, as we do not have a ground truth to compare against in this case, and because this comparison is not perfect. There are some differences in the simulations that have to be taken into account. Most notably, USSim considers the ambient lighting as a function of D65 (a standard daylight spectrum), whereas ours is based on a white

spectrum (which would be more fitting for a cloudy day). Besides, they use slightly modified medium coefficients. We have tried to bring ours as close as possible, but the conditions are still not identical.

However, the comparison is still useful to give a broad idea of the outputs of both methods. As we can see in the mentioned figure (Figure 6.6), both pipelines are able to produce similar results under similar conditions. We can see the added quality of our single scattering implementation, which adds volumetric shadows in the middle of the image on the $depth = 3$ m scenes, as well as the caustics. USSim, on the other hand, adds the godrays as particle effects (e.g., in the Jerlov IA, $depth = 9$ m scene), which do not achieve the same level of realism and controllability.

Their implementation is however faster (around 100 fps, instead of our 35 fps when using Single Scattering). Part of our cost is due to our hyperspectral simulation, but most of it comes from the volumetric lighting, as will be discussed in the computational cost section 6.3.

In the background of the Jerlov IA scene, we see another limitation of USSim. They consider the color at infinity to be equal to their ambient light A. In reality, there may be absorption events in between (especially in absorption dominated waters such as this one), resulting in a much lower radiance, as we can see in our renders.

Their caustics are implemented in texture space, so they do not properly interact with the geometry of the scene. On the other hand, our method projects them from the light source, resulting in a more realistic simulation.

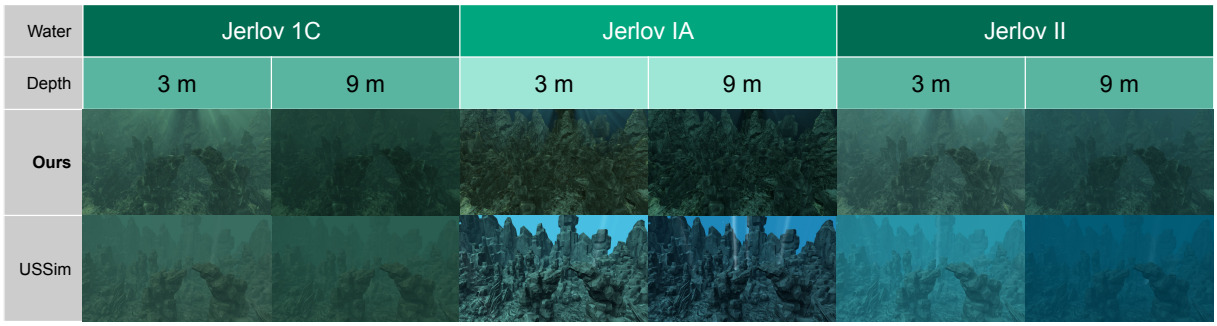


Figure 6.6: Comparison between our method and USSim, on three media, at two different depths each.

In conclusion, although both methods account for a similar approximation of ambient illumination, our model has the advantage of being an analytical solution to the VRE. Most notably, this allows it to simulate the appearance of different depths, even at different points of the same scene, whereas USSim uses its single ambient illumination for the whole area.

6.3 Computational cost

The computational cost of our method has been measured for a range of numbers of wavelength bands between $n = 3$ up to $n = 1000$. In practice, there are early diminishing returns in the accuracy of hyperspectral rendering, so 1000 wavelengths would not be used in any practical application. However, measuring the frame rendering time for that number of wavelengths is

interesting for the study of the computational cost.

In theory, the rendering time of the algorithm for n wavelengths should be:

$$t_{frame}(n) = t_{overhead} + t_{SS} + nt_{MS} \tag{6.1}$$

where $t_{overhead}$ is the time spent in overhead due to data transfer to the GPU and other computations (such as postprocessing effects); t_{SS} is the cost of Single Scattering approximation, and t_{MS} the time to evaluate a single wavelength of our multiple scattering method (Equation (4.7)).

As only the multiple scattering computations are affected by the number of wavelengths, the order of the algorithm is O(n) with respect to this parameter.

In Figure 6.7, we can see the empirical results of the experiment. The rendering times have been measured in a consumer laptop with an NVIDIA GeForce GTX 1050 GPU. Single scattering is enabled, and the shown rendering times are in seconds.

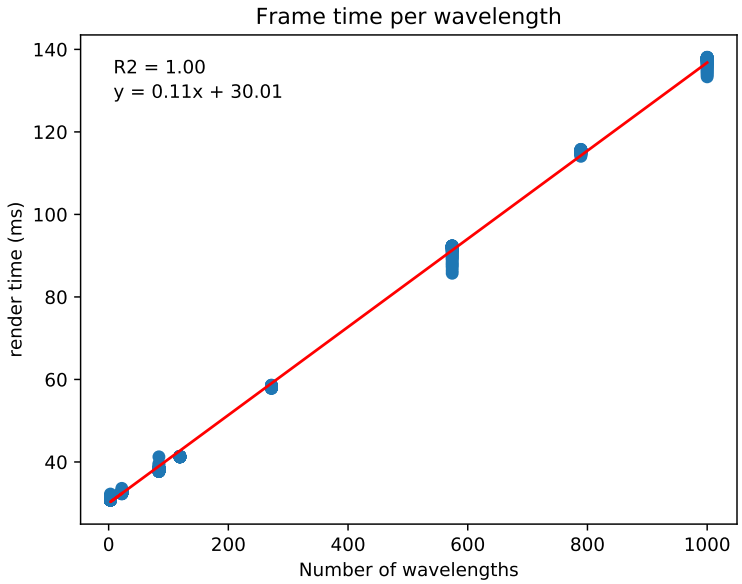


Figure 6.7: Plot of the rendering time (ms) with respect to the number of wavelengths evaluated.

As we can see, the relationship is indeed linear, and the rendering time (in ms) can approximately be predicted by the regressed function $t(n) = 0.1068n + 30.01$.

As we can see, for low values of n, the overhead due to other operations besides our proposed spectral rendering method is responsible for most of the rendering time. The computational cost of our approximation only becomes more noticeable around $n = 100$.

It could be possible to greatly accelerate the implementation for greater n values if it was required, especially leveraging techniques such as SIMD (Single Instruction Multiple Data) or other forms of parallelism. However, in this case we do not believe it is necessary, as a large portion of the spectral resolution that would be gained would be lost when converting to RGB

for the display (see section 4.2).

7. Conclusions

We have proposed a two part method to render the two main components of underwater lighting:

- Low frequency multiple scattering.
- High frequency single scattering.

We have validated our hyperspectral approximation for multiple scattering considering a scene under diffuse lighting (equivalent to a cloudy day) against a path traced simulation. Qualitatively, the results suggest that our approximation works well for these kind of scenes, for all the Jerlov waters. These water types show great variety: they can be coastal (1C, 3C...), open waters (I, II, III), and turbid (9C) or clear (1C). We have also carried out a more quantitative evaluation of the importance of spectral rendering, comparing our method to an RGB implementation of our approximation. The spectral version significantly outperforms this implementation in all the evaluated error metrics, and its computational cost is only slightly higher.

Meanwhile, our single scattering pipeline is less physically based – it needs to prioritize performance to work in real time. As we only consider the lighting that gets past the surface of the water, we are able to directly emulate the caustics patterns without computing expensive refractions that form them in the real world. This implementation allows the caustics to light surfaces as well as the volumetric component.

This full pipeline results in a physically-based, flexible simulation able to react to a great variety of physical parameters such as depth, water types, the sensor response curve, sun radiance, etc.

We have integrated our method into an interactive application showing an underwater scene. We have also added a user interface to control most of the parameters in real time. The test functionalities used throughout this report have also been implemented into the application.

Our method is not without its limitations. To enable our analytical solution we have made several approximations. We assume a homogeneous medium (the coefficients of absorption and scattering do not depend on the position). In reality, the concentrations of the biological and inorganic constituents that modify these coefficients do vary across the water bodies. This variation is however low, and even more so on greater depths [CBS*18].

We also assume that the phase function of the water is isotropic to simplify our integral of multiple scattering. Often, a Mie phase function is used [BG73; ZRP74], which models the

distribution of scattering directions assuming the medium is composed of homogeneous spherical particles. Other works propose even more accurate models [ODB*18]. However, we believe that our assumption is fair in our case, especially combined with our omnidirectional ambient light. If the lighting was more directional, the phase function would play a more important role in the final radiance result.

Tackling these limitations would be an interesting research direction in the future. Other future work could arise from expanding the scope of our project. For example, we did not focus on the appearance of the water surface, which is a rich and complex feature in oceans. As we mentioned in the related work section (1.1), the simulation and imitation of waves has led to large amounts of works. However, few focus on its appearance from underwater.

An interesting effect to render, related to refraction and total internal reflection underwater, is Snell's window [HV95; Lyn15], in which the whole sky hemisphere is projected to a circle of a certain angle in the water surface.

In conclusion, both our method and our proposed UWRP pipeline can be easily extended. As our model has been formally derived from the physics of light transport and oceanography, it can represent a sound foundation for further development in the area of real-time underwater appearance synthesis. In the technical side, the proposed pipeline could also be built upon, as it can work seamlessly with other features of game engines. Therefore, this work can represent the first step towards a more complete real-time underwater simulation framework.

Bibliography

[AK90] Arvo, James and Kirk, David B. "Particle transport and image synthesis". *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990) 7.

[Ame22] Ameye, Alexander. *Rendering realtime caustics*. May 2022. URL: https://alexanderameye.github.io/notes/realtime-caustics/ 30.

[AT18] Akkaynak, Derya and Treibitz, Tali. "A Revised Underwater Image Formation Model". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, 6723–6732. DOI: 10.1109/CVPR.2018.00703 5, 7, 9.

[BG73] Brown, Otis B and Gordon, Howard R. "Two component Mie scattering models of Sargasso Sea particles". *Applied optics* 12.10 (1973), 2461–2465 40.

[Blo19] Blough, Neil V. "Photochemical Processes". *Encyclopedia of Ocean Sciences (Third Edition)*. Ed. by Cochran, J. Kirk, Bokuniewicz, Henry J., and Yager, Patricia L. Third Edition. Oxford: Academic Press, 2019, 39–48. ISBN: 978-0-12-813082-7. DOI: https://doi.org/10.1016/B978-0-12-409548-9.11607-0. URL: https://www.sciencedirect.com/science/article/pii/B9780124095489116070 6, 17.

[CBMC14] Cronin, Thomas W, Bok, Michael J, Marshall, N Justin, and Caldwell, Roy L. "Filtering and polychromatic vision in mantis shrimps: themes in visible and ultraviolet vision". English. *Philosophical Transactions of the Royal Society B: Biological Sciences* 369.1636 (2014). ISSN: 1471-2970. DOI: 10.1098/rstb.2013.0032 5.

[CBS*18] Costello, Mark John, Basher, Zeenatul, Sayre, Roger, et al. "Stratifying ocean sampling globally and with depth to account for environmental variability". *Scientific Reports* 8.1 (July 2018). DOI: 10.1038/s41598-018-29419-1. URL: https://doi.org/10.1038/s41598-018-29419-1 40.

[Cha50] Chandrasekhar, Subrahmanyan. *Radiative transfer*. 1950 7, 10.

[CZSZ22] Chen, Kaixin, Zhang, Lin, Shen, Ying, and Zhou, Yicong. "Towards Controllable and Physical Interpretable Underwater Scene Simulation". *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, 2789–2793. DOI: 10.1109/ICASSP43922.2022.9747308 9, 36.

[Dev16] Developer, Game. *How Subnautica plunges deeper into rendering realistic water*. Feb. 2016. URL: https://www.gamedeveloper.com/design/how-i-subnautica-i-plunges-deeper-into-rendering-realistic-water 8.

[DWWH20] Deng, Hong, Wang, Beibei, Wang, Rui, and Holzschuch, Nicolas. "A practical path guiding method for participating media". *Computational Visual Media* 6.1 (2020), 37–51 7.

[EAJ05] Ernst, Manfred, Akenine-Möller, Tomas, and Jensen, Henrik. "Interactive rendering of caustics using interpolated warped volumes". Jan. 2005, 87–96. DOI: 10.1145/1089508.1089523 28.

[GJJD09] Gutierrez, Diego, Jensen, Henrik Wann, Jarosz, Wojciech, and Donner, Craig. "Scattering". *SIGGRAPH ASIA Courses*. ACM, 2009 4.

[GKH*13] Georgiev, Iliyan, Křivánek, Jaroslav, Hachisuka, Toshiya, et al. "Joint importance sampling of low-order volumetric scattering". en. *ACM Trans. Graph.* 32.6 (Nov. 2013), 1–14 7.

[GLF*08] Gutierrez, Diego, Lopez-Moreno, Jorge, Fandos, Jorge, et al. "Depicting Procedural Caustics in Single Images". *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 27.5 (2008), 120:1–120:9 28.

[Gor89] Gordon, Howard R. "Can the Lambert-Beer law be applied to the diffuse attenuation coefficient of ocean water?": *Limnology and Oceanography* 34.8 (1989), 1389–1409. DOI: https://doi.org/10.4319/lo.1989.34.8.1389. eprint: https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lo.1989.34.8.1389. URL: https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lo.1989.34.8.1389 6, 17.

[GSMA08] Gutierrez, Diego, Seron, Francisco, Muñoz, Adolfo, and Anson, Oscar. "Visualizing Underwater Ocean Optics". *Computer Graphics Forum (Proc. of EUROGRAPHICS)* 27.2 (2008), 547–556 5, 7.

[Hil15] Hillaire, Sébastien. *Physically-based & Unified Volumetric Rendering in Frostbite*. July 2015 26.

[HV95] Horváth, Gábor and Varjú, Dezsö. "Underwater refraction-polarization patterns of skylight perceived by aquatic animals through Snell's window of the flat water surface". *Vision Research* 35.12 (June 1995), 1651–1666. doi: 10.1016/0042-6989(94)00254-j. url: https://doi.org/10.1016/0042-6989(94)00254-j 41.

[HZE*19] Herholz, Sebastian, Zhao, Yangyang, Elek, Oskar, et al. "Volume path guiding based on zero-variance random walk theory". en. *ACM Trans. Graph.* 38.3 (June 2019), 1–19 7.

[IDN02] Iwasaki, Kei, Dobashi, Yoshinori, and Nishita, Tomoyuki. "An Efficient Method for Rendering Underwater Optical Effects Using Graphics Hardware". *Computer Graphics Forum* 21 (Dec. 2002), 1–11. doi: 10.1111/1467-8659.00628 28.

[IDN03] Iwasaki, Kei, Dobashi, Yoshinori, and Nishita, Tomoyuki. "A Fast Rendering Method for Refractive and Reflective Caustics Due to Water Surfaces". *Computer Graphics Forum* 22.3 (2003), 601–609. doi: https://doi.org/10.1111/1467-8659.t01-2-00708. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.t01-2-00708. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.t01-2-00708 28.

[JA18] Jarabo, Adrian and Arellano, Victor. "Bidirectional rendering of vector light transport". *Computer Graphics Forum*. Vol. 37. 6. Wiley Online Library. 2018, 96–105 7.

[JC98] Jensen, Henrik Wann and Christensen, Per H. "Efficient simulation of light transport in scences with participating media using photon maps". *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. Not Known: ACM Press, 1998 7.

[JDZJ08] Jarosz, Wojciech, Donner, Craig, Zwicker, Matthias, and Jensen, Henrik Wann. "Radiance caching for participating media". en. *ACM Trans. Graph.* 27.1 (Mar. 2008), 1–11 7.

[Jen95] Jensen, Henrik Wann. "Importance driven path tracing using the photon map". *Eurographics*. Vienna: Springer Vienna, 1995, 326–335 7.

[JNSJ11] Jarosz, Wojciech, Nowrouzezahrai, Derek, Sadeghi, Iman, and Jensen, Henrik Wann. "A comprehensive theory of volumetric radiance estimation using photon points and beams". en. *ACM Trans. Graph.* 30.1 (Jan. 2011), 1–19 7.

[JNT*11] Jarosz, Wojciech, Nowrouzezahrai, Derek, Thomas, Robert, et al. "Progressive photon beams". en. *ACM Trans. Graph.* 30.6 (Dec. 2011), 1–12 7.

[JW15] Jeschke, Stefan and Wojtan, Chris. "Water wave animation via wavefront parameter interpolation". *ACM Transactions on Graphics (TOG)* 34.3 (2015), 1–14 7.

[JW17] Jeschke, Stefan and Wojtan, Chris. "Water wave packets". *ACM Transactions on Graphics (TOG)* 36.4 (2017), 1–12 7.

[JZJ08] Jarosz, Wojciech, Zwicker, Matthias, and Jensen, Henrik Wann. "The beam radiance estimate for volumetric photon mapping". en. *Comput. Graph. Forum* 27.2 (Aug. 2008), 557–566 7.

[Kaj86] Kajiya, James T. "The Rendering Equation". *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986, 143–150. isbn: 0897911962. doi: 10.1145/15922.15902. url: https://doi.org/10.1145/15922.15902 7, 10.

[Kd14] Křivánek, Jaroslav and d'Eon, Eugene. "A zero-variance-based sampling scheme for Monte Carlo subsurface scattering". *ACM SIGGRAPH 2014 Talks on - SIGGRAPH '14*. Vancouver, Canada: ACM Press, 2014 7.

[KGH*14] Křivánek, Jaroslav, Georgiev, Iliyan, Hachisuka, Toshiya, et al. "Unifying points, beams, and paths in volumetric light transport simulation". en. *ACM Trans. Graph.* 33.4 (July 2014), 1–13 7.

[Kov20] Kovalovs, Artem. "Volumetric Effects of The Last of Us: Part Two". *ACM SIGGRAPH 2020 Talks*. SIGGRAPH '20. Virtual Event, USA: Association for Computing Machinery, 2020. isbn: 9781450379717. doi: 10.1145/3388767.3407393. url: https://doi.org/10.1145/3388767.3407393 26.

[KTT13] Kim, Theodore, Tessendorf, Jerry, and Thuerey, Nils. "Closest point turbulence for liquid surfaces". *ACM Transactions on Graphics (TOG)* 32.2 (2013), 1–13 7.

[Lag18] Lagarde, Sebastièn. *The Road toward Unified Rendering with Unity's High Definition Render Pipeline*. Aug. 2018. url: http://advances.realtimerendering.com/s2018/index.htm 28.

[LDC*05] Lee, Zhong-Ping, Darecki, Miroslaw, Carder, Kendall L., et al. "Diffuse attenuation coefficient of downwelling irradiance: An evaluation of remote sensing methods". *Journal of Geophysical Research: Oceans* 110.C2 (2005). doi: https://doi.org/10.1029/2004JC002573 6, 17.

[LKA*17] Liarokapis, Fotis, Kouřil, P., Agrafiotis, Panagiotis, et al. "3D MODELLING AND MAPPING FOR VIRTUAL EXPLORATION OF UNDERWATER ARCHAEOLOGY ASSETS". *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W3 (Feb. 2017), 425–431. doi: 10.5194/isprs-archives-XLII-2-W3-425-2017 8.

[LW96] Lafortune, Eric P and Willems, Yves D. "Rendering participating media with bidirectional path tracing". *Eurographics*. Vienna: Springer Vienna, 1996, 91–100 7.

[LX13] Liu, Shiguang and Xiong, Yuan. "Fast and stable simulation of virtual water scenes with interactions". *Virtual Reality* 17.1 (2013), 77–88 7.

[Lyn15] Lynch, David K. "Snell's window in wavy water". *Appl. Opt.* 54.4 (Feb. 2015), B8–B11. doi: 10.1364/AO.54.0000B8. url: https://opg.optica.org/ao/abstract.cfm?URI=ao-54-4-B8 41.

[MGN17] Müller, Thomas, Gross, Markus, and Novák, Jan. "Practical path guiding for efficient light-transport simulation". en. *Comput. Graph. Forum* 36.4 (July 2017), 91–100 7.

[MHD16] Meng, Johannes, Hanika, Johannes, and Dachsbacher, Carsten. "Improving the Dwivedi sampling scheme". en. *Comput. Graph. Forum* 35.4 (July 2016), 37–44 7.

[MJJG18] Marco, Julio, Jarabo, Adrian, Jarosz, Wojciech, and Gutierrez, Diego. "Second-order occlusion-aware volumetric radiance caching". en. *ACM Trans. Graph.* 37.2 (July 2018), 1–14 7.

[MMR*19] Müller, Thomas, Mcwilliams, Brian, Rousselle, Fabrice, et al. "Neural importance sampling". en. *ACM Trans. Graph.* 38.5 (Oct. 2019), 1–19 7.

[Mob01] Mobley, C.D. "Radiative Transfer in the Ocean*". *Encyclopedia of Ocean Sciences (Second Edition)*. Ed. by Steele, John H. Second Edition. Oxford: Academic Press, 2001, 619–628. isbn: 978-0-12-374473-9. doi: https://doi.org/10.1016/B978-012374473-9.00469-0. url: https://www.sciencedirect.com/science/article/pii/B9780123744739004690 6, 17.

[Mob94] Mobley, Curtis. *Light and Water: Radiative Transfer in Natural Waters*. Jan. 1994 6, 17.

[Muñ14] Muñoz, Adolfo. "Higher Order Ray Marching". *Computer Graphics Forum* 33.8 (2014), 167–176. issn: 1467-8659. doi: 10.1111/cgf.12424. url: http://dx.doi.org/10.1111/cgf.12424 12.

[NSB13] Nielsen, Michael B, Söderström, Andreas, and Bridson, Robert. "Synthesizing waves from animated height fields". *ACM Transactions on Graphics (TOG)* 32.1 (2013), 1–9 7.

[ODB*18] Organelli, Emanuele, Dall'Olmo, Giorgio, Brewin, Robert J. W., et al. "The open-ocean missing backscattering is in the structural complexity of particles". *Nature Communications* 9.1 (Dec. 2018). doi: 10.1038/s41467-018-07814-6. url: https://doi.org/10.1038/s41467-018-07814-6 41.

[PA00] Premoze, S. and Ashikhmin, M. "Rendering natural waters". *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*. 2000, 23–434. doi: 10.1109/PCCGA.2000.883856 7.

[PKK00] Pauly, Mark, Kollig, Thomas, and Keller, Alexander. "Metropolis light transport for participating media". *Eurographics*. Vienna: Springer Vienna, 2000, 11–22 7.

[PM93] Pattanaik, S N and Mudur, S P. "Computation of global illumination in a participating medium by monte carlo simulation". en. *J. Vis. Comput. Animat.* 4.3 (July 1993), 133–152 7.

[PP20] Park, Sekil and Park, Jinah. "Realistic simulation of mixed sea using multiple spectrum-based wave systems". *Simulation* 96.3 (2020), 281–296 7.

[RPC89] Roesler, Collin S., Perry, Mary Jane, and Carder, Kendall L. "Modeling in situ phytoplankton absorption from total absorption spectra in productive inland marine waters". *Limnology and Oceanography* 34.8 (1989), 1510–1523. doi: https://doi.org/10.4319/lo.1989.34.8.1510. eprint: https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lo.1989.34.8.1510. url: https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lo.1989.34.8.1510 6.

[RSSF02] Reinhard, Erik, Stark, Michael, Shirley, Peter, and Ferwerda, James. "Photographic Tone Reproduction for Digital Images". *ACM Trans. Graph.* 21.3 (July 2002), 267–276. issn: 0730-0301. doi: 10.1145/566654.566575. url: https://doi.org/10.1145/566654.566575 24.

[SK04] Schechner, Y.Y. and Karpel, N. "Clear underwater vision". *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. 2004, I–I. doi: 10.1109/CVPR.2004.1315078 6.

[SM15] Solonenko, Michael G. and Mobley, Curtis D. "Inherent optical properties of Jerlov water types". *Appl. Opt.* 54.17 (June 2015), 5392–5401. doi: 10.1364/AO.54.005392. url: https://opg.optica.org/ao/abstract.cfm?URI=ao-54-17-5392 4, 6, 31.

[Sta96] Stam, Jos. "Random caustics: natural textures and wave theory revisited". (Jan. 1996). doi: 10.1145/253607.253883 5, 30.

[TCR19] Thompson, Stephen, Chalmers, Andrew, and Rhee, Taehyun. "Real-Time Mixed Reality Rendering for Underwater 360° Videos". *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2019, 74–82. doi: 10.1109/ISMAR.2019.00-24 8.

[VG97] Veach, Eric and Guibas, Leonidas J. "Metropolis light transport". *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*. Not Known: ACM Press, 1997 7.

[Vos14] Vos, Nathan. "Volumetric Light Effects in Killzone: Shadow Fall". *GPU Pro 5*. Ed. by Engel, Wolfgang. CRC Press, 2014, 127–147 5, 26.

[Wat90] Watt, Mark. "Light-Water Interaction Using Backward Beam Tracing". *SIGGRAPH Comput. Graph.* 24.4 (Sept. 1990), 377–385. issn: 0097-8930. doi: 10.1145/97880.97920. url: https://doi.org/10.1145/97880.97920 28.

[ZRP74] Zaneveld, J Ronald V, Roach, David M, and Pak, Hasong. "The determination of the index of refraction distribution of oceanic particulates". *Journal of Geophysical Research* 79.27 (1974), 4091–4095 40.