Dynamic Ocean Inverse Modeling based on Differentiable Rendering

Xueguang Xie
SKL of VR Tech. & Syst., Beihang Univ.
Beijing, 100191, China
Qingdao RI, Beihang Univ.
Qingdao, 266100, China

Yang Gao
SKL of VR Tech. & Syst., Beihang Univ.
Beijing, 100191, China

Fei Hou
SKLCS, Institute of Software,
Chinese Academy of Sciences
Beijing, 100190, China
Univ. of Chinese Academy of Sciences
Beijing, 100049, China
houfei@ios.ac.cn

Aimin Hao
SKL of VR Tech. & Syst.,
Beihang Univ.
Beijing, 100191, China
Peng Cheng Lab,
Shenzhen, 518000, China

Hong Qin
Computer Science Dept.,
Stony Brook Univ.
11794, USA

Abstract

Learning and inferring underlying motion patterns of captured 2D scenes and then re-creating dynamic evolution consistent with the real-world natural phenomena have high appeal for graphics and animation. To bridge the technical gap between virtual and real environments, we focus on the inverse modeling and reconstruction of visually consistent and property-verifiable oceans, which inherits the advantages of deep learning and differentiable physics to learn geometry and constituting waves in a self-supervised manner. First, we infer the hierarchical geometry using two networks, which are optimized via the differentiable renderer. Second, we extract the wave components from a series of inferred geometry through a network equipped with a differentiable ocean model. Then, the ocean dynamics can be evolved by the reconstructed wave components. Through extensive experiments, we verify that our new method yields satisfactory results for both geometry reconstruction and wave estimation. Moreover, this newly proposed approach outperforms the SOTA methods. The new framework has the inverse modeling potential to facilitate a host of graphics applications, such as the rapid production of physics-accurate scene animation and its re-editing guided by real ocean scenes.

Keywords: Inverse Modeling, Surface Reconstruction, Wave Estimation, Ocean Animation, Differentiable Rendering.

1. Introduction and Motivation

In recent years, realistic ocean simulation and its photorealistic rendering have achieved great success in simulators, movies, and video games. However, they are insufficient for the most complex VR/AR scenarios and the digital-twin modeling of natural phenomena, which rely heavily on real-world scene reconstruction, prediction, evolution, and physical re-editing. For example, video or geometry-based end-to-end fluid predictions contradict physics and lack realism due to the fluid's changeable geometry. Hence, real-world ocean reproduction and physics-guided re-editing in virtual environments remain urgent research priorities. In the real world, the ocean surface motion is always simulated via a sum of wave trains. To estimate model-based deformation rather than directly deforming 3D shape, we would like to reconstruct not only ocean geometry but also the component wave trains for model-based deformation.

Meanwhile, geometry reconstruction from images or videos has recently been improved by merging a deep learning network with its inverse process, differentiable rendering, which has been demonstrated to achieve considerable performance using a self-supervised training method. Hence, in this study, we aim to propose a deep learning and differentiable physics coupled framework for dynamic inverse modeling from real ocean videos including geometry reconstruction and component wave estimation, as shown in Figure 1. With the convincing visually and property-consistent reconstruction, our work can enhance the virtual-real fusion simulation of ocean-related natural phenomena.

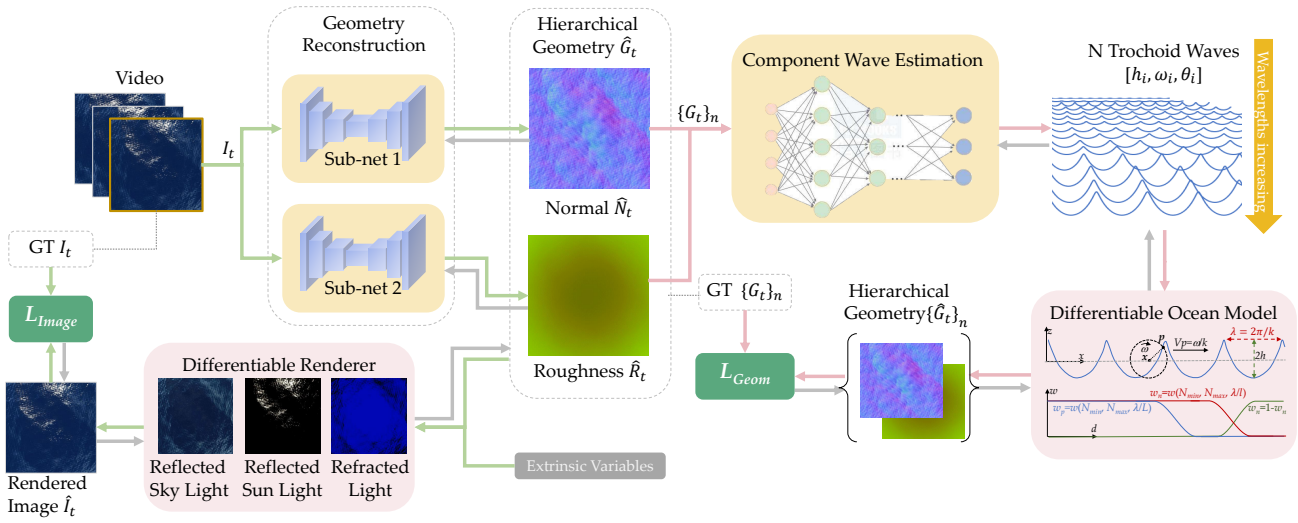

Figure 1. The overview of our framework. Given a sequence of monocular images, we use two encoder-decoder networks to reconstruct the hierarchical ocean geometry G_t for each image I_t. The hierarchical geometry includes the macrosurface normal N_t and the microfacet roughness R_t corresponding to the two sub-networks, respectively. Then we can acquire a sequence of ocean hierarchical geometry G_{tn}. Moreover, we use an MLP network to estimate the wave spectrum from the reconstructed geometry sequences. Each wave is represented by a set of parameters $\{h_i, \omega_i, \theta_i\}$. These networks are trained by combining differentiable rendering and differentiable ocean modeling as loss functions.

Particularly, we concentrate on the water surface of calm deep ocean alone, ignoring huge waves or waves on the coasts and shores.

The core of our algorithm is the wave trains estimation by self-supervised deep learning networks. Some oceanography studies have estimated wave spectrum on deep oceans using data captured by buoy motion sensors [26] or SAR images [39] and analyzed the frequency from height data, Such as Pierson-Moskowitz spectrum [30] and Jonswap spectrum [12]. Rather than the complicated acquired data, we intend to utilize common images to reconstruct their wave spectrum for deep oceans. Therefore, the extraction of geometry from images is an essential task. For ocean animation, researches commonly use three-hierarchical geometry to model surface, including the displacement, the normal of macrosurface, and the roughness of microfacet. Under a fixed perspective, large-amplitude waves influence displacement, large-amplitude and middle-amplitude waves affect the normal of geometry, and small-amplitude waves determine the microfacet (roughness). We adapt the above considerations for ocean simulation and rendering in the backward process. However, in the forward process, we just need to reconstruct two hierarchies of them and use them to infer component wave trains, i.e., normal and roughness, which can satisfy both re-rendering and inferring wave trains.

In summary, our main contributions include:

- **Inverse modeling of the hierarchical ocean geometry from videos.** We propose a self-supervised deep learning framework to reconstruct hierarchical geometry from ocean images with differentiable ocean rendering. The hierarchical geometry contains the normal of the macrosurface at the pixel level and the roughness of the microfacet at the sub-pixel level. Moreover, we utilize another deep learning network to estimate the wave trains from the sequence of the inferred hierarchical geometry with a differentiable ocean model as loss function.

- **Model-driven dynamic ocean generation by representing the surface with wave trains.** Based on the estimated ocean wave parameters, we can generate hierarchical geometry at different view distances through the ocean model, which provides a model-driven method for evolving, predicting, and re-editing the dynamic ocean.

- **New scenes synthesis based on differentiable renderer.** Our physical ocean renderer supports synthesizing variable view images and new lighting scenes. Due to its differentiability, the renderer can also be utilized for image-based parameter fitting, including camera parameters, ambient light, and sea colors. Hence, image-based ocean style transfer can be supported.

Our paper is organized as follows. We review the most related works of our study in the next section. Then we overview our algorithm in Section 3 and present the details of dynamic ocean inverse modeling in Section 4. In addition, we show our experiments and results in Section 5. And in Section 6, several expanded applications using our framework are presented. Finally, we conclude and explore avenues of future work in Section 7.

2. Related Work

Our work is closely relevant to ocean modeling and rendering, differentiable rendering and 3D reconstruction. We briefly review them in the following categories.

2.1. Ocean Surface Modeling and Rendering

Ocean modeling in computer graphics mainly includes physical fluid simulation, wave geometric modeling, and wave spectrum-based modeling. The first kind of approach simulates the sea surface based on the Navier-Stokes equations, which calculate in detail the motion of water particles. This ocean dynamic are clear and realistic, but the computation is complex [19, 15]. The wave shape modeling methods consider that ocean surface travel by a series of waves and the shape of each wave can be described as different mathematical equations. The theory developed from early sinusoids [24] to later Peachy model [29] and Gerstner model or trochoids proposed by Fournier and Reeves [7]. The Gerstner model solves the Euler fluid equations for gravity waves in deep water exactly. The advantage of this method is that the wave model is simple, the solution of the mathematical function is easy to obtain, and the ocean wave model is similar to the real scenes. However, due to the simple mathematical function, the generated wave models are solitary, weakly random, and only ideal for calm sea surface simulation. In oceanography, empirical wave spectrum approaches are proposed by aggregating the observed real ocean into mathematical distributions. The main wave spectrums include Pierson-Moskowitz spectrum, Jonswap spectrum, generalized A, B spectrum, Tessendorf spectrum and TMA Spectrum, etc. The Pierson-Moskowitz spectrum gives the energy distribution of gravity waves as a function of their frequency. Hasselmann *et al*. [12] proposed the Jonswap spectrum by multiplying an extra peak enhancement factor on the Pierson-Moskowitz spectrum to improve the fit for different scenes, then extended this model with a wave direction parameters [11]. The research [3] fitted the TMA spectral form to over 2,800 wind sea spectra to test its viability and to determine if any parametric relationships could be established linking the spectral parameters to the external wind field. These methods are costly while improving real-time and realism. Bruneton *et al*. [4] and Podee *et al*. [31] combined them together, i.e., using wave spectrum and wave geometry equations to simulate the ocean surface when low amplitude. They usually simulate the breakers or splashes generated by the waves of higher amplitude by using physical methods [15]. researchers focus on adding interaction details on the large open ocean surface through the combination of physical models. Huang *et al*. [17] simulated ships, splashes, and waves on a vast ocean by designing a hybrid time-stepping algorithm that combined a FLIP domain and an adaptively remeshed Boundary Element Method (BEM) domain for the incompressible Euler equations. Xiong *et al*. [42] modeled complex vortex-interface interaction problems on the free surface by combining the expressiveness of the Clebsch wave function in describing vortical structures with the level-set function's capacity to follow interfacial dynamics. By analyzing the above methods, we combine the expanded three-dimensional trochoids wave and the ocean wave spectrum to model the ocean surface animation. Meanwhile, we implement the process as a differentiable ocean model.

Researches in ocean rendering can be dated back to the work of Fournier and Reeves. Tessendorf *et al*. [36] presented a sophisticated lighting model for realistic reproduction of ocean waves, while Ashikhmin and Premože *et al*. [1, 32] presented a light transport approach for the complex lighting effects of the ocean. However, traditional physically-based rendering is time-consuming and impractical for real-time applications. Hu *et al*. [16], Schneider and Westermann [35] studied real-time water surface rendering. Bruneton *et al*. [4] accelerated the rendering with high performance hardware and incorporated more sophisticated lighting models, such as reflection, refraction, and Fresnel terms. Podee *et al*. [31] approximated the reflection direction distribution for the water surface as an elliptical Gaussian distribution and integrated the reflection contribution throughout the rendering interval time to avoid reflection aliasing and flickering. While these methods use a pipeline to render images, they do not meet the requirement of differentiability. Hence, we would like to combine differentiable renderings with sophisticated lighting ocean rendering to realize our differentiable ocean renderer.

2.2. Differentiable Rendering

Differentiable rendering differentiates the rendering operation, allowing it to be utilized to optimize parameters or reconstruct scenes from observations. Existing approaches can be roughly divided into two categories: mesh rasterization based rendering and ray tracer-based rendering. For example, OpenDR [23], DiRT [14], and Neural Mesh Renderer (NMR) [18] manually defined approximated gradients of the rendering operation to alter the position of the face. In contrast, SoftRasterizer [22] and DIB-R [5] redefined the rasterization as a continuous and differentiable function, allowing gradients to be computed automatically. Forrester *et al*. [6] first sampled the surface using non-differentiable

rasterization, then applied differentiable, depth-aware point splatting to produce the final image. For ray tracer-based or volume-based differentiable rendering, Li *et al*. [21] represented each 3D point as a multivariate Gaussian and performed occlusion reasoning with grid discretization and ray tracing. Such methods require an explicit volume to perform occlusion reasoning. Mitsuba 2 [27] are proposed by developing differentiable rendering for implicit surface representations with a focus on reconstructing rigid objects. Inspired by them, we aim to adopt the SoftRasterizer method to make our ocean rendering model differentiable due to the ocean surface represented by mesh.

2.3. 3D Reconstruction

The last few years have clearly shown the effectiveness of neural networks for 2D and 3D reasoning [8, 10]. At first, most 3D reconstruction methods rely on supervised training methods and costly annotations, which makes it challenging to collect all properties of 3D observations. Xie *et al*. [40] reconstructed ocean surface normal by an image-to-image translation network to cycle translate images and 2D surface normal maps. Recently, there has been a surge in efforts to integrate graphics rendering processes into neural network pipelines. Specifically, the common 3D self-supervision pipeline is applied by integrating the rendering layer to the predicted scene parameters, then applying the loss by comparing the rendered and input image in various ways. Liu *et al*. in SoftRasterizer method and Chen *et al*. in DIB-R method predicted the mesh of objects by using differentiable rendering. Li *et al*. [21] and Nimier-David *et al*. [27] reconstructed the volume of objects by combining ray tracer-based DR. More researches are being conducted in a variety of applications, such as image-based training of 3D object reconstruction [38], human pose estimation [28], hand pose estimation [2], and face reconstruction. NeRF [25] reconstructed a 3D geometry model using an implicit representation. For fluid reconstruction, Thapa *et al*. [37] reconstructed the dynamic fluid surface using a deep neural network and a differentiable renderer that renders the refractive distortion of an underwater reference background image. Li *et al*. [20] inversely modeled the gas parameters and reconstructed high-resolution scenes. Qiu *et al*. [33] used a conditional generative model to rapidly reconstruct the temporal density and velocity fields of gaseous phenomena based on the sequence of two projection views. Inspired by them, we will reconstruct the ocean surface using DR-based deep learning networks, whereas we anticipate surface normal and roughness simultaneously based on ocean surface characteristics.

3. Algorithmic Overview

The goal of our method is to reconstruct dynamic ocean models from videos for prediction or re-editing. Because the dynamic ocean is propelled by a sequence of waves that comprise the ocean wave spectrum, we estimate the component waves from the original data for dynamic ocean reconstruction. However, direct estimation is not feasible. Due to the causality between surface geometry and component waves, we adopt a two-step framework, first reconstructing the geometry of the ocean surface and then estimating wave parameters. In a wave spectrum, the wavelengths vary from a few millimeters to several hundred meters, and their frequencies also change accordingly. Some high-amplitude waves affect surface displacement, while very small-amplitude waves have a negligible impact on the surface displacement, which is with respect to the viewing distance. Some waves with high and middle amplitude have an impact on the macrosurface normal. While some minor waves affect the roughness of the microfacet. Thus, we reconstruct ocean surface geometry at two explicit and implicit levels, the normal of the macrosurface as well as the roughness of the microfacet, and estimate the whole component wave trains based on the two-layer geometry. Overall, to make realistic and vivid animations of the ocean surface, we need to recover accurate 3D geometry, including normal and roughness, and estimate implicit geometry representation, i.e., the component wave parameters.

Given a sequence of monocular images I_{tn}, we use two encoder-decoder neural networks to reconstruct the ocean hierarchical geometry G_t from each image I_t. The hierarchical geometry includes its surface normal N_t and roughness R_t, corresponding to the two sub-networks, respectively. Thus, we can acquire a sequence of ocean geometry G_{tn}. Moreover, we use a Multilayer Perceptron (MLP) network to estimate the wave trains from these geometric sequences. Each wave is represented by a set of parameters, including the amplitude h_i, the angular frequency ω_i, and the wave direction θ_i. Ocean rendering can be regarded as the inverse of geometry reconstruction, while ocean modeling by superposing several wave trains to generate the geometry is the inverse of component wave estimation. Hence, we train our networks for geometry reconstruction and wave estimation using a differentiable renderer and a differentiable ocean model as loss functions in a self-supervised way. In other words, we extract the hierarchical geometry from the image, then render it with known extrinsic variables into a new image $\hat{I}_t$ and ensure consistency between $\hat{I}_t$ and the input image I_t. Similarly, we estimate wave parameters using an MLP network and synthesize a time series of surface geometry using a differentiable ocean model. Then, we measure the consistency of the generated and the input geometry of the MLP network. The procedure is overviewed in Figure 1. The data generated by networks is marked by $\hat{}$ to distinguish it from ground truth, which is omitted without ambiguity.

4. Dynamic Ocean Inverse Modeling

4.1. Differentiable Ocean Model

Ocean surface motion is the superposition of numerous wave trains of different amplitudes, frequencies, directions, and chaotic phases. Each wave can be modeled by the Gerstner wave (trochoid), which is originally proposed by solving Euler fluid equations for gravity waves in deep water. In mathematics, the trochoid is the trajectory curve of a fixed point on the radius of a circle as the circle rolls down a specified line, as shown in Figure 2 (top). In this study, we model ocean surface waves using the extended 3D trochoid, as shown in Figure 3. And the trochoid equation in three dimensions is:

$$p(x,y,t) = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + \begin{bmatrix} h\cos\theta\sin(\omega t - \mathbf{k}\cdot\mathbf{X}) \\ h\sin\theta\sin(\omega t - \mathbf{k}\cdot\mathbf{X}) \\ h\cos(\omega t - \mathbf{k}\cdot\mathbf{X}) \end{bmatrix}, \quad (1)$$

where the displacement $\mathbf{p}$ in the plane coordinate of point $\mathbf{X}$ at time t is computed by the function $p(x,y,t)$, and $\mathbf{X} = \begin{bmatrix} x \\ y \end{bmatrix}$ is the ocean surface at rest. $h, \omega, \mathbf{k}$, and θ are the wave parameters involving amplitude, angular frequency, wave vector and wave direction, respectively. Actually, there is a certain internal relationship between these parameters, i.e., wave vector $\mathbf{k} = [k\cos\theta, k\sin\theta]$, and the wave number $k = \omega^2/g$. Therefore, we simplify the wave parameters from $\{h, \omega, \mathbf{k}, \theta\}$ to $\{h, \omega, \theta\}$, which can determine a trochoid shape of ocean waves.

The regular pattern of the infinite constituent wave trains is studied in oceanography by summarizing the observed real ocean into a mathematical distribution. Different empirical wave spectrums are modeled for different ocean scenes, such as Pierson-Moskowitz spectrum and Jonswap spectrum [12], as shown in Figure 4. The general spectrum formula is:

$$\begin{aligned} h &\propto \sqrt{S}, \\ S(\omega) &= \frac{\alpha g^2}{\omega^5}\exp\left[-\beta(\frac{\omega_0}{\omega})^4\right]\gamma^{\exp\left[-(\frac{\omega-\omega_0}{\sqrt{2}\sigma\omega_0})^2\right]}, \end{aligned} \quad (2)$$

where, $\alpha, \beta, \sigma, \gamma$, and ω_0 are parameters that determine the wave spectrum. When $\gamma = 1$, the formula is the Pierson-Moskowitz spectrum, otherwise it is the Jonswap spectrum. The amplitude of waves h is positively correlated with $\sqrt{S}$. Here, we simulate the ocean surface using the two spectrums, which reflect the hidden patterns of different component waves when wind speed is less than 25 km/h and the surface of the ocean does not produce whitecaps. Although various customized ocean spectrums may be utilized for specific phenomena, the two spectrums can handle the majority of wave instances and are rather efficient.

We simulate the ocean surface with discrete wave spectrums, where we sample N wave trains from the spectrum and express the parameter set of each wave as $\{h_i, \omega_i, \theta_i\}$. Then, we superimpose these surface waves. Actually, the details of surface geometry in ocean images are proportional to the viewing distance. Waves with wavelengths longer than the distance of the pixel can impact the macrosurface. Waves with a wavelength less than the distance of the pixel, such as at sub-pixel level, not only have no effect on surface height but also cause signal aliasing. In contrast, the sub-pixel wave affects the microfacet rather than the macrosurface. Therefore, we employ a Nyquist filter to calculate the weight of each wave being used, as shown in Figure 2(b). The blue line represents the weight to filter the long wave that affects displacement. The red line represents the weight to filter the middle and long waves that affect the normal of the macrosurface. And the green line represents the weight to filter the short wave that affects the roughness of the macrosurface. More details can be found in the Bruneton's work. Hence, we superpose the waves to generate hierarchical ocean geometry with the weight of the Nyquist filter.

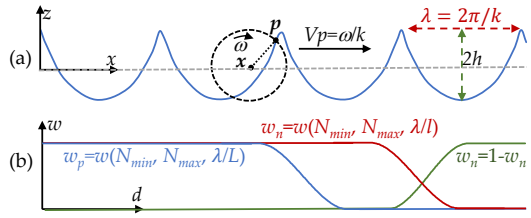

Figure 2. (a): 2D trochoid Waves formed by the trajectory curve of a fixed point on the radius of a circle as the circle rolls along a given line. The wave travels along the x-axis. The wave number is $k = \omega^2/g$. The wavelength is $\lambda = 2\pi/k$. The wave speed is $V_p = \omega/k$. The parameters, amplitude h and angular frequency ω, determine the shape of a trochoid for ocean waves; (b): The weight for three wave hierarchies using the Nyquist filter.

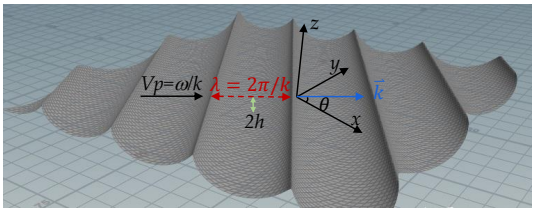

Figure 3. 3D trochoid waves. The wave propagates in the direction of the wave vector $\mathbf{k}$ in the x-y plane. θ denotes the angle between the vector $\mathbf{k}$ and the x-axis. $\mathbf{k} = [k\cos\theta, k\sin\theta]$, where k is the wave number. The parameters, amplitude h, angular frequency ω, and wave direction θ, determine the shape of a Gerstner surface wave.

The wave spectrum determines the characteristics of the ocean surface. As a result, given a sequence of surface geometry, constituent waves can be deduced, similar to the

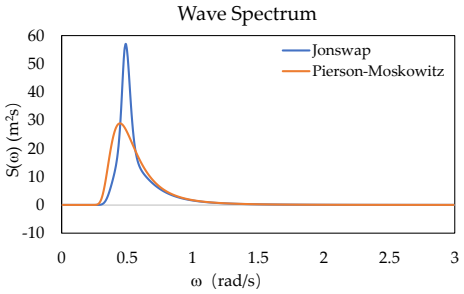

Figure 4. The ocean wave spectrum. It depicts the hidden patterns of various component waves in various scenes. The main wave spectrums include Pierson-Moskowitz spectrum and Jonswap spectrum.

wave spectrum estimation, which is the inverse of ocean surface modeling. In our framework, we use the model not only to simulate ocean datasets but also to estimate constituent waves by making the model differentiable, as described in Section 4.4.

4.2. Differentiable Ocean Rendering

The most recent ocean animation study portrays the ocean surface using hierarchical geometry, including displacement, surface normal, and roughness, which achieves good performances in both geometry simulation and realistic rendering. We adopt the three hierarchical models for ocean geometry to maintain more details on the surface. Specifically, we model the surface using a triangle mesh that is driven by large waves at the grid level for geometry deformation. Then, project and interpolate the grid into image space. For more details, we add more middle waves at the pixel level to the normal computed using the triangle mesh. Besides, the roughness of the microfacet in each pixel is estimated by measuring the slope variance of these little waves. The image can be rendered using the normal and roughness values for each pixel. When implementing the algorithm, instead of rasterizing in a traditional rendering pipeline, we use the SoftRas method in [22], i.e., a continuous and differentiable rasterization function. According to the preceding description, the computation of the entire hierarchical geometry can be completed. We consider the reflected light from the sun and sky dome, as well as the refracted light from the water. Overall, we compute displacements to alter the geometry and compute normals and roughness to render images by our differentiable ocean renderer under certain lighting conditions. The renderer is used to both generate the realistic images as a dataset and to be the loss to optimize the related networks. Given an ocean image and its corresponding geometry, the differentiable renderer can also be used to fit the camera and lighting information of the image by iterative optimization.

4.3. Geometry Reconstruction Network

Because the normal and roughness of the ocean surface are closely related to the ocean image during the rendering, we reconstruct them from the ocean photographs by merging a deep learning network with differentiable rendering. Specifically, given a single color image I_t as input, we use two Res-Net neural networks [13], sub-net1 and sub-net2, to reconstruct the ocean hierarchical geometry G_t, including surface normal N_t and roughness R_t. Then, a differentiable renderer is applied to render a new image $\hat{I}_t$ by inputting the normal N_t, the roughness R_t, and extrinsic variables. We also calculate the difference between the new re-rendered image $\hat{I}_t$ and the input image I_t. Due to the differentiability of our ocean renderer, the error gradient can be propagated back to the network parameters to optimize the networks. The loss function is defined as:

$$L_{Image} = \mathbb{E}_{I_t, \hat{I}_t}[\|I_t - \hat{I}_t\|_2]. \tag{3}$$

The normal map is a dense field with numerous details. In contrast, the roughness map has a relatively low frequency. And for the synthetic data, the real normal and roughness can be captured easily. Hence, we use a spatial discriminator D_n and a spatial discriminator D_r for surface normal and roughness to enforce that the inferred data spatial distribution is similar to the real data. We also use a temporal discriminator D_{tn} and D_{tr} to maintain temporal coherence. The spatial discriminators and the temporal discriminator reference the method [41]. As a result, the overall loss is defined as:

$$\begin{aligned} L_{Total} =& \alpha L_{Image} + \beta_1 L_{Dn} + \beta_2 L_{Dr} + \\ & \gamma_1 L_{Dtn} + \gamma_2 L_{Dtr}, \end{aligned} \tag{4}$$

where, $\alpha, \beta_1, \beta_2, \gamma_1$ and γ_2 are the weights of each part. $L_{Dn}, L_{Ds}, L_{Dtn}, L_{Dts}$ are the losses of the corresponding discriminators. In our experiment, we use the synthetic data and the total loss function to train the networks.

4.4. Wave Parameters Estimation Network

To reconstruct a dynamic ocean, we need to estimate the constituent waves from real-world images and videos. In the previous section, we reconstructed ocean geometry from images, including normal and roughness. The goal of this section is to estimate the parameters of various constituent waves based on the time sequence of the reconstructed ocean geometry. We use a UNet [34] to extract features from the series of geometries, and use an MLP network to extract wave parameters from these features. During training, we use our differentiable ocean model to generate a series of new geometries based on the extracted wave parameters and measure the consistency of the new geometry with the input. A cosine similarity loss is applied for

normal vector and an MSE loss is used for roughness. The formula for the cosine similarity loss function is:

$$L_{cos} = \mathbb{E}_{\mathbf{N}_t, \hat{\mathbf{N}}_t}[(1 - \cos(\mathbf{N}_t, \hat{\mathbf{N}}_t))]. \quad (5)$$

The formula of L_2 for roughness is defined as:

$$L_2 = \mathbb{E}_{R_t, \hat{R}_t}[\|R_t - \hat{R}_t\|_2]. \quad (6)$$

The total loss function for our wave estimation network includes two terms above, which is defined as:

$$L_{Geom} = \alpha L_2 + \beta L_{cos}, \quad (7)$$

where α and β are the weights of each part.

5. Experiments

5.1. Data Preparation and Network Training

We generate the dataset with our differentiable ocean modeling and ocean rendering algorithms, which are implemented in *Taichi (https://taichi.graphics/)* framework due to its automatic differentiation feature. Our dataset includes ocean image sequences, normal, roughness, and wave parameters, as well as camera position and lighting conditions. To ensure sufficient variation in physical motions and dynamics, we randomly sample the wave trains from various wave spectrums and use a random camera and lighting setups. For our dataset, we generate 2,000 scenes with different initial conditions and each scene includes 10 frames. In order to test the generalization ability to new scenes that did not appear in the training set, we randomly selected 200 completed scenes as the test set. To monitor model overfitting to determine the number of training epochs, we randomly divided the rest of the fragments into a training set and a validation set in a ratio of 9:1.

Our networks are implemented in Python 3.8 with PyTorch 1.11.0, and the differentiable ocean model and differentiable renderer are written in the *Taichi* framework. Our networks are optimized using Adam with $\varepsilon = 1e-8$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ on a single NVIDIA GeForce RTX 2080 SUPER GPU. We split the training process into two phases. Sub-net1 and sub-net2 are trained together with loss function L_{Total} for 200 epochs, We alternately train the four auxiliary discriminators and the generator, like typical GANs training [9]. The wave parameter estimation network is then trained for 100 epochs using the loss function L_{Geom} on data generated by sub-net1 and sub-net2.

5.2. Results and Evaluations

Geometry Reconstruction. We evaluate our network sub-net1 and sub-net2 of ocean geometry reconstruction on the testset. They are exhibited in Figure 5, including the input original image I_t, our reconstructed normal $\hat{N}_t$ and roughness $\hat{R}_t$, and the re-rendered image $\hat{I}_t$ based on our reconstructed results. Besides, ground truths and error maps are also provided. It can be seen that the reconstructed normal and roughness are close to the ground truth, and the re-rendered image is also comparable to the input image. The given error maps show that both our reconstructed results and the re-rendered image have low errors. Furthermore, we quantitatively evaluate our generated results using different error metrics. The quantitative evaluation results are shown in the first line of Table 1. We use the mean absolute error (MAE) for the normal in spherical coordinates, the cosine similarity (CosSim) for the normal in Cartesian coordinates, and the mean square error (MSE) for the roughness and MSE for the image. These error values are at a very low level. In addition, we calculate the percentage of error angle between our normal and ground truth within each range. The error angle within 10° is up to 78%, and more than 90% of normal is within 15°. We also compare our method to other variants that use UNet [34] or ResNet, and use different forms of normals as output (i.e., 3D vector in Cartesian coordinate or 2D in spherical coordinate). Their quantitative evaluation results are shown in Table 1. In comparison, we find that our framework using ResNet and the output form of 3D normal is the top performer.

Our network sub-net1 uses a ResNet for normal reconstruction, similar to the surface reconstruction network in the method [40], which combines surface reconstruction network with a neural rendering network for adversarial training. While we train our network using a differentiable renderer as a loss function. In addition, we add another structure for the roughness reconstruction of microfacets. Thanks to these improvements, the reconstruction result is significantly improved, and our framework supports more complicated re-editing such as varying camera position and varying lighting. The comparison is shown in Figure 6.

Wave Parameters Estimation. We test our wave parameter estimation network on the testset, and the results are shown in Figure 7. The results include 60 waves, each with three parameters, namely $\{h_i, \omega_i, \theta_i\}$. The left in Figure 7 shows the amplitudes of 60 waves, the middle is the angular frequency, and the right is θ representing wave spreading direction. In the left figure, the amplitudes in green are predicted by our network, while the amplitudes in blue are corrected by the wave spectrum, i.e., enforcing the frequency of component waves to fit the spectrum distribution and enforcing the frequency and amplitudes to be negatively correlated using Equation 2. The corrected results achieve better performances. Table 2 illustrates the quantitative evaluation for estimated parameters. We use the MAE and the MSE to measure the distance between estimated parameters and ground truth, both of which have low values. As shown in Figure 8, we compare the final synthesized geometry based on our estimated component waves to the geometry directly

Table 1. Quantitative comparisons of our results with ground truth

Method	Roughness	Image	Normal		Error Angle of Normal		
	MSE	MSE	MAE-SpereN	CosSim-VecN	Within 5°	Within 10°	Within15°
ResNet-VecN	0.0170	0.0161	0.1667	0.0113	38.12%	78.19%	93.38%
ResNet-SpereN	0.0169	0.0181	0.2394	0.0141	30.03%	70.57%	90.19%
UNet-VecN	0.0187	0.0331	0.2912	0.0323	27.76%	68.35%	87.98%

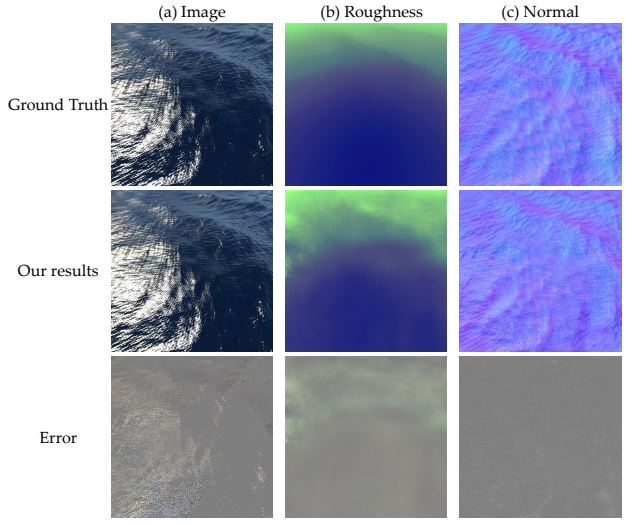

Figure 5. The results of generated geometry. Top: the input image and the ground truths of roughness and normal; Middle: Our re-rendered image and our reconstructed results of roughness and normal; Bottom: the error maps between our results and ground truths. Each column shows images, roughness and normal maps. Three normal components are mapped to the RGB color components.

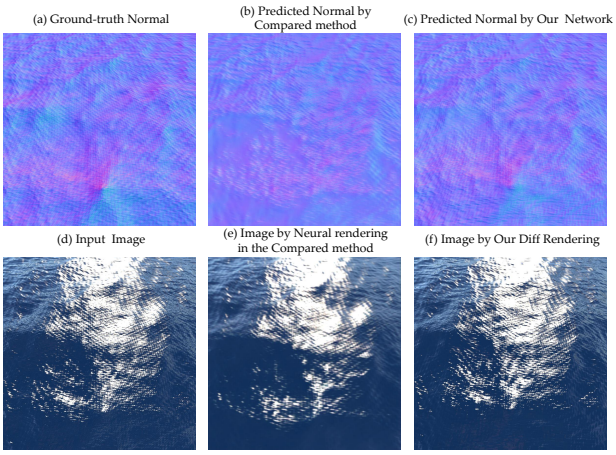

Figure 6. Comparisons of normal reconstruction and image re-rendering. Left: the input image (bottom) and the ground-truth normal(top); Middle: the reconstructed normal and re-rendered image by the compared method; Right: our reconstructed normal and re-rendered image. Our results are cleaner than the compared method because of using the differentiable renderer instead of neural rendering and reconstructing microfacets simultaneously.

reconstructed by our first-step networks, i.e., the intermediate results as well as the second step input. Both the normal and the roughness are extremely analogous. In some small areas, our final results maintain more details while the generated result by the network is fuzzy, as shown in the close-up of the normal. More results are illustrated in the supplementary video. The estimated wave parameters can be applied to synthesize new scenes that are property-consistent with the source ocean video.

6. Applications

Image-based Parameter Fitting. Our framework can extract imaging parameters from observed images, such as illumination, camera positions, and sea colors. First, the surface normal and roughness can be reconstructed using the methods described above. Then the lighting parameters can be optimized iteratively by the differentiable renderer. In other words, we use our ocean renderer to generate a new image with the reconstructed normal, roughness, and a series of initial parameters, then match the rendered results with the input original image. Due to our renderer's differentiability, the gradient of the lighting parameters can be calculated using gradient propagation. Thus, the parameters can be optimized incrementally. The lighting parameters include the color of sky light, the color of Sun light and the direction of Sun light. The optimization is run for 400 iterations using Adam. We show the convergence plot of loss and various parameters during iterations in Figure 9. The loss has been reduced to $1e-6$ and all the optimized parameters have converged to their ground truths. To be more intuitive, we compare the rendered images with the fitted parameters and the random initial parameters to the ground truth. The results are given in Figure 10. The extracted imaging parameters can be used for ocean-style transfer and rendering homologous scenes. Our final experiments of ocean re-rendering in Figure 14(c) are based on the estimated illumination, camera positions, and sea colors by this method.

Geometry-based Wave Parameters Fitting. In the above framework, we use the network to estimate the parameters of component waves, which uses a differentiable ocean surface model as the loss to train the network. In addition, the parameters of component waves can also be estimated via iterative optimization by the differentiable ocean

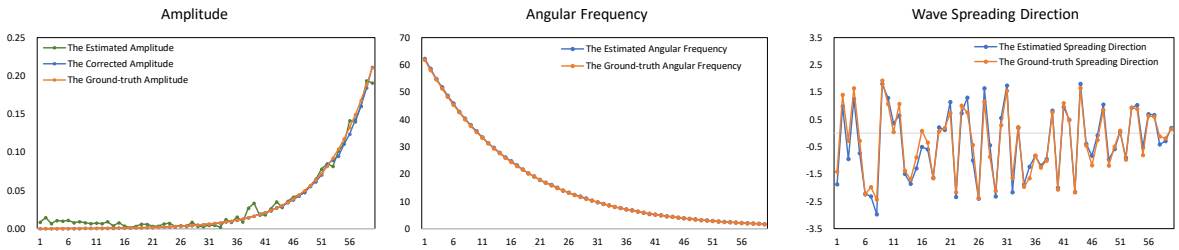

Figure 7. Results of parameter estimation. Left: the amplitude of each wave. The parameters in green are predicted by the network, while the amplitudes in blue are corrected by the wave spectrum, which achieves better performance. Middle: our estimated angular frequencies versus the ground truth. Right: our estimated θ against the ground truth. The estimated parameters (in blue) are close to the ground truth (in yellow).

Table 2. Quantitative measurements of estimated wave parameters against the ground truth

Method	Amplitude	Angular Frequency	Spreading Direction
MSE	0.00004	0.03522	0.08431
MAE	0.00518	0.13100	0.22691

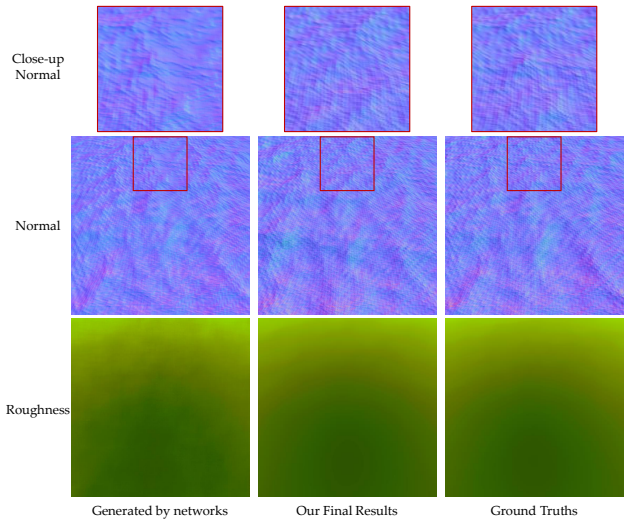

Figure 8. Comparisons of the geometry directly reconstructed by the network with the geometry synthesized based on the estimated wave parameters. They are extremely similar. In some parts, the former are blurry while our final results maintain more details.

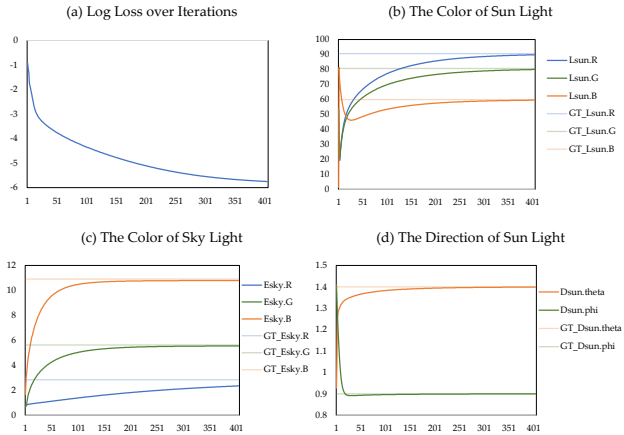

Figure 9. Convergence of lighting parameters. (a): the logarithmic error with the number of iterations. (b-d): the color of sunlight, the color of sky light, and the direction of sunlight, respectively. The parameters (in dark colors) converge to the ground truths (in light colors) after several iterations.

model. Specifically, the hierarchical geometry is modeled with a set of randomly initialized waves, which are then matched to the real geometry to optimize the wave parameters. Thanks to the differentiable ocean modeling approach, our fitted geometry can converge towards the ground truths, as shown in Figure 11. Figure 12 shows the parameter optimization of four waves.

Animation Prediction and Ocean Scene Re-editing. We can reconstruct and forecast the dynamic ocean using our framework, which includes component wave parameter estimation, ocean surface advection, and ocean rendering and re-editing depending on the predicted parameters. Figure 13 shows the reconstructed and predicted ocean based on estimated wave parameters. The predicted dynamic ocean is realistic, and our model-driven technique can forecast ocean animation over a long period of time with minimal error. Moreover, our framework supports scene re-editing, including changing camera and lighting parameters. Figure 15 shows an example of re-editing the scene on synthetic data. (a) is the re-simulated results with the lighting $Dsun = [1.0, -1.5]$ and camera $c\theta = 1.5, ch = 10$. (b) and (c) are the re-edited scenes with changed camera parameters, and (d) has the changed lighting parameters.

We also apply our framework to real-world ocean scenarios. Figure 14 displays the reconstructed ocean and re-edited scenes with varying ambient lighting. The reconstructed images in (c) are based on the reconstructed nor-

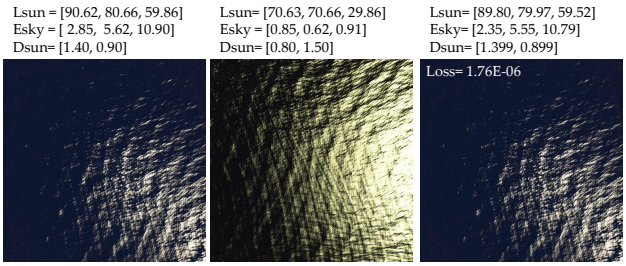

Figure 10. The results of lighting reasoning. From left to right: the rendered images with ground truth lighting, random initial lighting, and the estimated lighting, respectively. The value of each parameter is displayed above each image. The images rendered by our predicted lighting and the ground truths are visually indistinguishable. The error is six orders of magnitude negative.

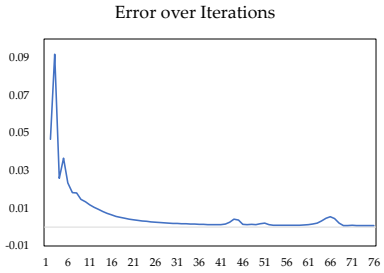

Figure 11. Optimization of wave parameters. The loss gradually converges to zero after several iterations.

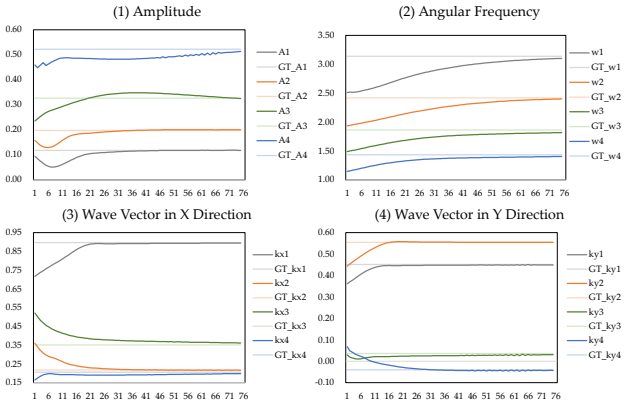

Figure 12. Optimization of wave parameters. Each figure shows a plot of the fitted wave parameters for four waves (lines in dark color) over the number of iterations. Our results converge to their ground-truth wave parameters (lines in a light color) gradually. Each wave includes four parameters of amplitude, angular frequency, wave vector in x direction, and wave vector in y direction.

mals in (b) and the estimated illumination, camera positions, and sea colors by our method. The small error in (d) between our reconstructed images and the input images illustrates the effectiveness of our framework for both reconstructing geometry and estimating imaging parameters. The re-editing scene in (e) demonstrates our framework provides the possibility to create more novel scenes with varying lighting. Furthermore, we show another example of re-editing scenes with varying camera parameters in Figure 16 for this real scene. Our method can replicate the ocean motion trends while generating more diverse scenery. More results of ocean dynamic inverse modding and re-editing are shown in the supplemental video.

Limitations. Our networks for reconstructing geometry have high generality for ocean scenes, while our proposed model is based on the deep water waves model of Pierson-Moskowitz spectrum and Jonswap spectrum, which are good at deep water waves but can hardly work for the coasts and shores. Hence, our algorithm shares the spectrum's constraints. That is to say, restricted by the wave model, the network for wave estimation can only cover calm deep oceans. For more complicated scenes, *e.g.* rough oceans or large-scale interactions between the ocean and solid objects immersed or floating in it, the wave spectrum cannot be estimated. Furthermore, the scenes do not contain whitecaps, which develop when wind speeds exceed 25 km/h. There is a reason to assume that the reconstruction of ocean scenes with whitecaps can be implied using the same framework with different renderers and ocean spectrums. Furthermore, our experiments rely on the synthesis data, and future work will include capturing ocean images and working with real data. We did not consider shadowing cases because, for distant views, its effects are already taken into account in the Ross BRDF in our renderer. Although we did not specifically consider self-shadowing for close views, if such scenes exist in our dataset, our data-driven model can learn how to deal with them automatically.

which mentioned that self-shadowing or shadowing from other objects can be provided with a shadow map for close views, which for distance views its effects are already taken into account in the Ross BRDF. In our dataset, we did not consider shadowing cases. If these scenes exist in our dataset, our data-driven model can be automatically learned how to deal with it.

7. Discussion and Conclusion

To reproduce a dynamic ocean based on real ocean video in a virtual environment, this study has detailed a revolutionary ocean inverse modeling method from images that takes advantage of deep learning, differentiable rendering, and differentiable ocean model. To drive the ocean motion, we not only reconstruct the ocean surface hierarchical geometry from images but also estimate the component waves of the surface geometry. Specifically, we design two sub-networks to extract the normal of the macrosurface and the roughness of the microfacet and train them in a self-supervised manner with a differentiable ocean renderer as a loss function. Afterward, we use an MLP network to esti-

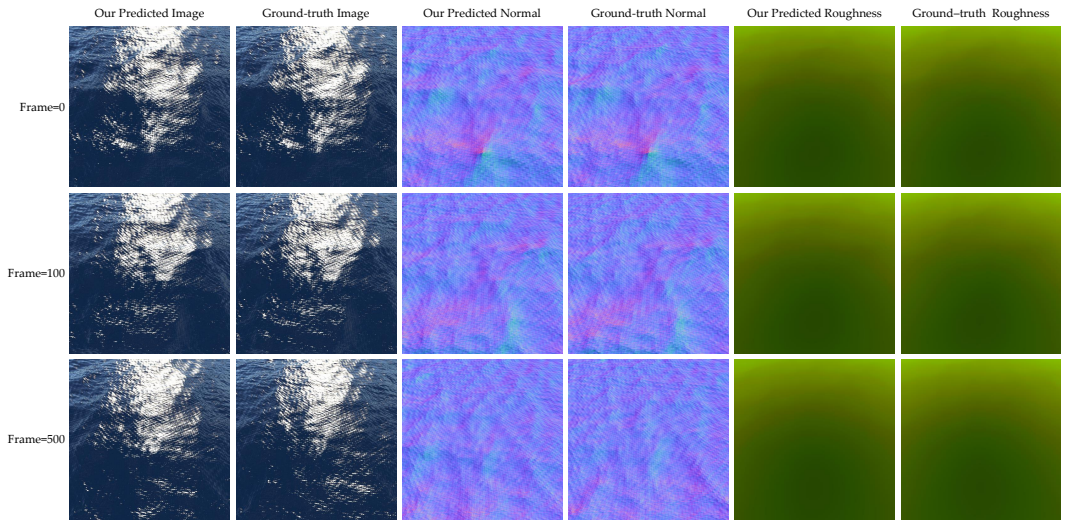

Figure 13. Reconstruction and prediction of the ocean. The first two columns are our predicted and ground-truth images. The middle two columns display our predicted normal maps and ground truths. The last two columns show our predicted roughness and ground truths. We show the results at frames 0, 100, and 500, respectively.

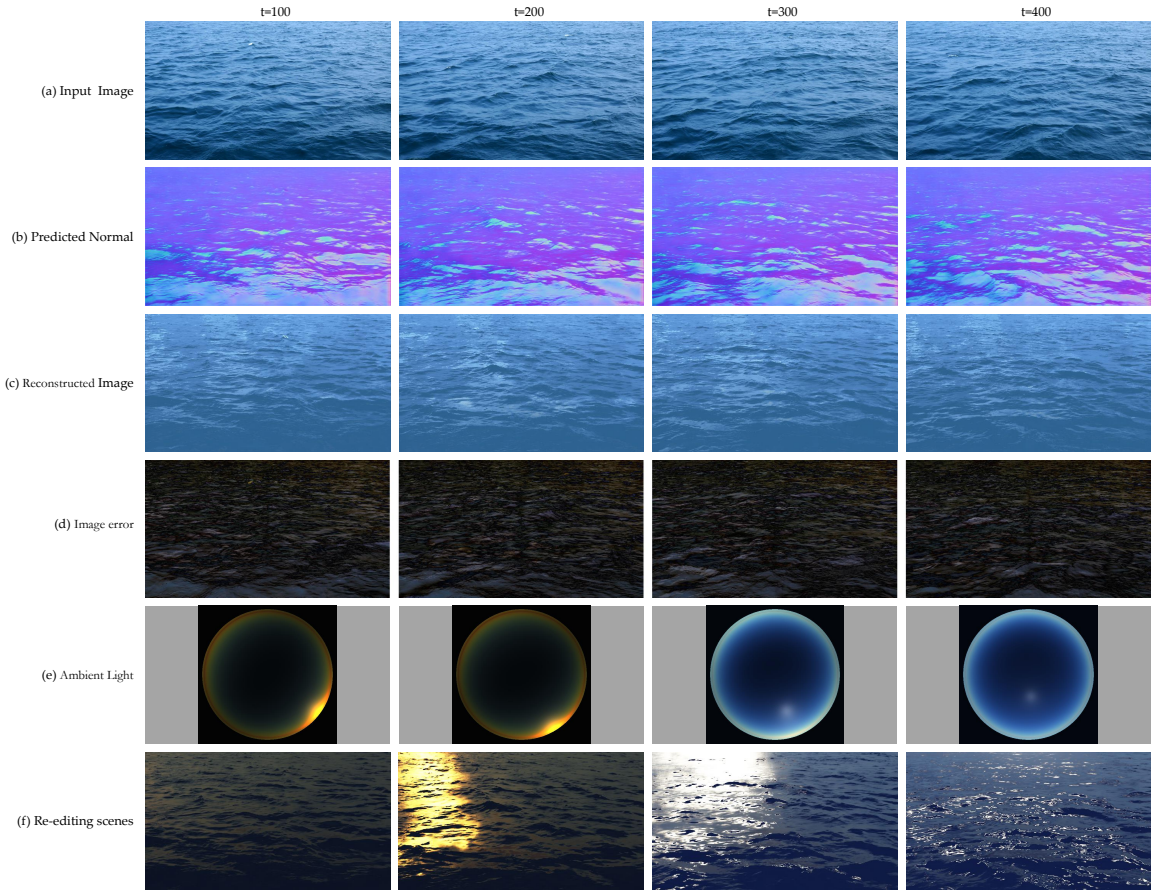

Figure 14. Real scene reconstruction and re-editing. From top to bottom: the captured real scenes, reconstructed normal, reconstructed scenes, image error, reference ambient light and the novel synthetic scenes by re-editing using corresponding light. We show the results at frames 100, 200, 300, and 400, respectively.

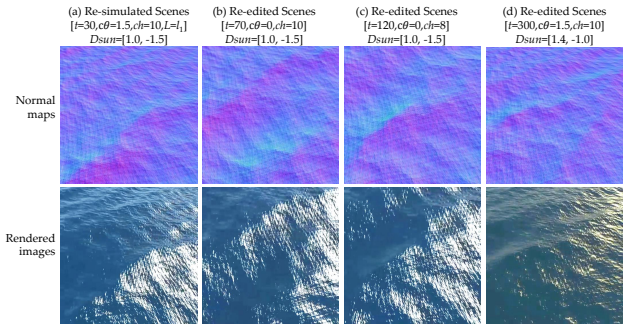

Figure 15. Re-editing the scene by changing imaging parameters, including the camera view $c\theta$, camera height ch, and direction of sunlight $Dsun$. (a): the re-simulated scene at $t = 30$ with $c\theta = 1.5, ch = 10, Dsun = [1.0, -1.5]$; (b): the re-edited scene at $t = 70$ with $c\theta = 0, ch = 10$; (c): the re-edited scene at $t = 120$ with $c\theta = 0, ch = 8$; (d): the re-edited scene at $t = 300$ with $Dsun = [1.4, -1.0]$.

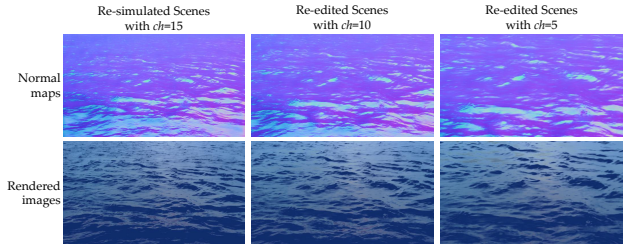

Figure 16. Re-editing the real scene by changing camera parameters. From left to right: the camera height decreases, i.e., from the initial value $ch = 15$ to the re-edited value $ch = 10$ and $ch = 5$.

mate component wave parameters from a sequence of hierarchical geometry, using the differentiable ocean model as a loss function. The dynamic ocean can be reconstructed and forecasted by using approximated waves. Besides, the ocean scenes can be re-edited flexibly, such as by transferring illumination and varying views. Our differentiable renderer and ocean model can be used individually to fit the lighting parameters from images or fit wave parameters from geometry sequence. The experiment results have proved that our framework has exhibited effectiveness and usability for dynamic ocean reconstruction.

It would be worthwhile to investigate whether our strategy may be extended to more general applications. Moreover, we intend to explore a novel method by combining other sophisticated models, such as shallow water equations or differentiable physical fluid simulation, to reconstruct more scenarios in the future.

Acknowledgements

This work was sponsored by the National Natural Science Foundation of China under Grants (62002010, 61872347), Huawei Innovation Research Plan (TC20220616019), also in part by the US National Science Foundation (IIS-1715985 and IIS-1812606), and the CAMS Innovation Fund for Medical Sciences (CIFMS) under Grant 2019-I2M-5-016, Special Plan for the Development of Distinguished Young Scientists of ISCAS (Y8RC535018).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] M. Ashikhmin, S. Premoze, P. Shirley, and B. E. Smits. A variance analysis of the metropolis light transport algorithm. *Computers & Graphics*, 25(2):287–294, 2001. 3

[2] S. Baek, K. I. Kim, and T. K. Kim. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. In *Conference on Computer Vision and Pattern Recognition*, pages 1067–1076, 2019. 4

[3] E. Bouws, H. Günther, W. Rosenthal, and C. Vincent. Similarity of the wind wave spectrum in finite depth water: 1. spectral form. *Journal of Geophysical Research: Oceans*, 90(C1):975–986, 1985. 3

[4] E. Bruneton, F. Neyret, and N. Holzschuch. Real-time realistic ocean lighting using seamless transitions from geometry to BRDF. *Computer Graphics Forum*, 29(2):487–496, 2010. 3

[5] W. Chen, J. Gao, H. Ling, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances In Neural Information Processing Systems*, pages 9609–9619, 2019. 3

[6] F. Cole, K. Genova, A. Sud, D. Vlasic, and Z. Zhang. Differentiable surface rendering via non-differentiable sampling. In *International Conference on Computer Vision*, pages 6088–6097, 2021. 3

[7] A. Fournier and W. T. Reeves. A simple model of ocean waves. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pages 75–84, 1986. 3

[8] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499, 2016. 4

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, pages 2672–2680, 2014. 7

[10] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. In *International Conference on 3D Vision*, pages 412–420, 2017. 4

[11] D. E. Hasselmann, M. Dunckel, and J. Ewing. Directional wave spectra observed during jonswap 1973. *Journal of Physical Oceanography*, 10(8):1264–1280, 1980. 3

[12] K. Hasselmann, T. Barnett, E. Bouws, H. Carlson, D. Cartwright, K. Enke, J. Ewing, H. Gienapp, D. Hasselmann, P. Kruseman, A. Meerburg, P. Muller, D. Olbers,

K. Richter, W. Sell, and H. Walden. Measurements of wind-wave growth and swell decay during the joint north sea wave project (jonswap). *Deut. Hydrogr. Z.*, 8:1–95, 01 1973. 2, 3, 5

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6

[14] P. Henderson and V. Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision*, 128(4):835–854, 2020. 3

[15] R. Hopper and K. Wolter. The water effects of pirates of the caribbean: Dead men tell no tales. In *ACM SIGGRAPH Talks*, pages 1–2, 2017. 3

[16] Y. Hu, L. Velho, X. Tong, B. Guo, and H. Shum. Realistic, real-time rendering of ocean waves. *Journal of Visualization and Computer Animation*, 17(1):59–67, 2006. 3

[17] L. Huang, Z. Qu, X. Tan, X. Zhang, D. L. Michels, and C. Jiang. Ships, splashes, and waves on a vast ocean. *ACM Transactions on Graphics*, 40(6):1–15, 2021. 3

[18] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 3

[19] A. T. Layton and M. van de Panne. A numerically efficient and stable algorithm for animating water waves. *The Visual Computer*, 18(1):41–53, 2002. 3

[20] C. Li, S. Qiu, C. Wang, and H. Qin. Learning physical parameters and detail enhancement for gaseous scene design based on data guidance. *IEEE Transactions on Visualization and Computer Graphics*, 27(10):3867–3880, 2020. 4

[21] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics*, 37(6):1–11, 2018. 4

[22] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *International Conference on Computer Vision*, pages 7708–7717, 2019. 3, 6

[23] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169, 2014. 3

[24] N. L. Max. Vectorized procedural models for natural terrain: Waves and islands in the sunset. *ACM SIGGRAPH Computer Graphics*, 15(3):317–324, 1981. 3

[25] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 4

[26] U. D. Nielsen and J. Dietz. Ocean wave spectrum estimation using measured vessel motions from an in-service container ship. *Marine Structures*, 69:102682, 2020. 2

[27] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics*, 38(6):1–17, 2019. 4

[28] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018. 4

[29] D. R. Peachey. Modeling waves and surf. *ACM SIGGRAPH Computer Graphics*, 20(4):65–74, 1986. 3

[30] W. J. Pierson and L. Moskowitz. A proposed spectral form for fully developed wind seas based on the similarity theory of s. a. kitaigorodskii. *Journal of Geophysical Research Atmospheres*, 69(24):5181–5190, 1964. 2

[31] N. Podee, N. Max, K. Iwasaki, and Y. Dobashi. Temporal and spatial anti-aliasing for rendering reflections on water waves. *Computational Visual Media*, 7(2):201–215, 2021. 3

[32] S. Premoze and M. Ashikhmin. Rendering natural waters. *Computer Graphics Forum*, 20(4):189–199, 2001. 3

[33] S. Qiu, C. Li, C. Wang, and H. Qin. A rapid, end-to-end, generative model for gaseous phenomena from limited views. In *Computer Graphics Forum*, pages 242–257, 2021. 4

[34] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241, 2015. 6, 7

[35] J. Schneider and R. Westermann. Towards real-time visual simulation of water surfaces. In *Proceedings of the Vision Modeling and Visualization Conference*, pages 211–218, 2001. 3

[36] J. Tessendorf. Simulating ocean water. *Simulating Nature: Realistic and Interactive Techniques Course Notes on SIGGRAPH*, pages 3:1–3:26, 2001. 3

[37] S. Thapa, N. Li, and J. Ye. Dynamic fluid surface reconstruction using deep neural network. In *Conference on Computer Vision and Pattern Recognition*, pages 21–30, 2020. 4

[38] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Conference on Computer Vision and Pattern Recognition*, pages 2626–2634, 2017. 4

[39] S. Vasavi, C. Divya, and A. S. Sarma. Detection of solitary ocean internal waves from sar images by using u-net and kdv solver technique. *Global Transitions Proceedings*, 2(2):145–151, 2021. 2

[40] X. Xie, X. Zhai, F. Hou, A. Hao, and H. Qin. Multitask learning on monocular water images: Surface reconstruction and image synthesis. *Computer Animation and Virtual Worlds*, 30(3-4):e1896:1–e1896:13, 2019. 4, 7

[41] Y. Xie, E. Franz, M. Chu, and N. Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics*, 37(4):712–726, 2018. 6

[42] S. Xiong, Z. Wang, M. Wang, and B. Zhu. A clebsch method for free-surface vortical flow simulation. *ACM Transactions on Graphics*, 41(4):116:1–116:13, 2022. 3