OpenWaters: Photorealistic Simulations For Underwater Computer Vision

Mehdi Mousavi
smousavi2@student.gsu.edu
Georgia State University
Atlanta, Georgia, USA

Shardul Vaidya
svaidya2@student.gsu.edu
Georgia State University
Atlanta, Georgia, USA

Razat Sutradhar
rsutradhar1@student.gsu.edu
Georgia State University
Atlanta, Georgia, USA

Ashwin Ashok
aashok@gsu.edu
Georgia State University
Atlanta, Georgia, USA

ABSTRACT

In this paper, we present OpenWaters, a real-time open-source underwater simulation kit for generating photorealistic underwater scenes. OpenWaters supports creation of massive amount of underwater images by emulating diverse real-world conditions. It allows for fine controls over every variable in a simulation instance, including geometry, rendering parameters like ray-traced water caustics, scattering, and ground-truth labels. Using underwater depth (distance between camera and object) estimation as the use-case, we showcase and validate the capabilities of OpenWaters to model underwater scenes that are used to train a deep neural network for depth estimation. Our experimental evaluation demonstrates depth estimation using synthetic underwater images with high accuracy, and feasibility of transfer-learning of features from synthetic to real-world images.

ACM Reference Format:
Mehdi Mousavi, Shardul Vaidya, Razat Sutradhar, and Ashwin Ashok. 2021. OpenWaters: Photorealistic Simulations For Underwater Computer Vision . In *The 15th International Conference on Underwater Networks and Systems (WUWNet'21), November 22–24, 2021, Shenzhen, Guangdong, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3491315.3491336

1 INTRODUCTION

Computer vision can turn noticeably more difficult when applied to under water images. Light transport in a dense medium like water creates phenomena such as caustics, refraction, dense scattering and absorption. Availability of large volume datasets has enabled deep learning methods to achieve state-of-the-art results in computer vision depth estimation. However, these datasets are immutable, and modification of imaging characteristics is not possible after acquisition.

Manual labeling of depth in underwater images is very challenging, so one is limited to active depth sensing (e.g., RGB-D sensors) or passive methods like disparity matching. Gathering data from underwater sources often requires expensive, delicate equipment that need to be enclosed in watertight containers, which can particularly affect depth estimation cameras [3], by blocking depth sensing rays in active depth sensing cameras; or making it difficult to track points in passive stereo vision cameras. All assuming that there is access to a controlled, physical underwater setup (i.e. a pool) to gather images from. As such, underwater vision problems rarely have large amounts of labelled data with accurate ground-truth annotations.

OpenWaters. To address the fundamental limitations in data collection for underwater computer vision, in this paper, we have designed OpenWaters , an underwater simulation kit designed to be highly customizable, easily extendable, run in real-time and open-source. An OpenWaters simulation can be shaped for any underwater visual task, including visible light communications, depth estimation and image enhancement. OpenWaters is capable of generating dynamic, massive image datasets in highly complex scenarios. As we detail further in section 3, OpenWaters is a modular set of customizable abstract classes developed within NVIDIA's RTX Branch of Unreal Engine 4 (Epic Games, USA)[8, 18]. OpenWaters is built with compatibility and low entrance barrier in mind, allowing even those researchers who are unfamiliar with Unreal Engine to create, maintain and iterate on their own photorealistic customized datasets for any vision task underwater. OpenWaters contains three key modules: (1) A *stochastic scene generation* system that creates the geometry of a scene based on parameters like objects, objectives, camera type, lighting, and backgrounds; (2) A *ground truth core* that automatically calculates and overlays a wide variety of accurate, pixelwise ground-truth annotations; (3) A *data ablation core* that enables changing characteristics of the simulation (i.e water, environment, lighting, timescale or rendering) in real-time without affecting scene composition.

In summary, the key contributions of this paper are as follows:

(1) Design of a simulation tool system for underwater camera scene generation emulating real-world conditions.
(2) A convolutional neural network (CNN) deep learning model for underwater depth estimation using OpenWaters data.
(3) Experimentation data trace based evaluation of depth estimation accuracy using OpenWaters synthetic and real-world data.

WUWNet'21, November 22–24, 2021, Shenzhen, Guangdong, China

ACM ISBN 978-1-4503-9562-5/21/11…$15.00
https://doi.org/10.1145/3491315.3491336

2 RELATED WORK

Underwater Simulations. Currently, there are open-source underwater simulations available. Prior simulations exclusively developed for underwater uses are UWSim (published in 2012) [20] and UUV (Unmanned Underwater Vehicle) Simulator (published in 2016) [14]. These tools generally allow for visualization of virtual underwater scenarios, and provide simulations of rigid body dynamics or sensors such as sonar or pressure sensors. However, these simulation tools haven't been recently updated or developed to support modern hardware. More importantly, these simulations do not focus on real-time, realistic image rendering with hardware-accelerated ray tracing, nor they are designed for modern diagnostic methods such as data ablation. [16, 26]. In comparison to other simulation tools, our system — OpenWaters — uses hardware ray tracing to accurately generate Photorealistic images in real time, its designed to be user friendly, and has integrated tools for data ablation experiments.

Underwater Depth Estimation. Scattering and absorption make underwater depth estimation more challenging in comparison to on-land scenarios[6, 19]. Prior work on Depth estimation has produced state-of-the-art results for in-land datasets such as NYUv2 or DIODE-Depth [7, 24]. Using deep learning, pixelwise accurate measurements can be achieved that all but eliminate the need for active depth sensing or stereo cameras for indoor scenes [2, 16]. The indoor data-sets gathered with active depth sensing cameras can use the structure of the scene to infer a pixel-wise depth map. Although these laser cameras work on indoor or outdoor scenes, submerging them underwater will interfere with their depth sensing capabilities [3]. This has led to studies using unsupervised methods for underwater depth estimation [10], or bringing images from other domains (Style transfer) using methods such as Generative Adversarial Networks (GANs), so they resemble underwater images [11, 12]. Generally, these studies seem to be hampered by the lack of reliable, flexible data [11, 25].

Using Synthetic Data. Modern computer graphics can achieve near-photorealism, so synthetic data has become a viable alternative in situations where acquiring or labeling real data is difficult. Synthetic data has been proven to be useful in complex computer vision tasks, such as depth estimation [16, 21, 22], surface normal estimation, robotic grasping [13, 22] and object segmentation [15] by multiple independent researchers [9, 16, 22].

3 OPENWATERS SIMULATION KIT

OpenWaters is a set of customizable classes, made in Unreal Engine for generating massive computer vision datasets in underwater scenarios. Unreal Engine 4 (UE4) is a graphics engine for projects with high-resolution, real-time 3D graphics. It is free for both commercial and non-commercial use and its source code is publicly available and extended by the community. We use one such extended version created for hardware accelerated ray-tracing by NVIDIA RTX graphics cards [18].

The key contributions of OpenWaters are: (i) Open-source, free set of user-friendly, extendable tools for creating customized datasets of underwater scenarios, (ii) A set of curated, customizable photorealistic scenes for underwater visual communications using real-time hardware ray-tracing, (iii) Extendable, Automatic, multi-modal, accurate ground-truth generation for visual tasks, and (iv) Compatible Python scripts for data acquisition and processing.

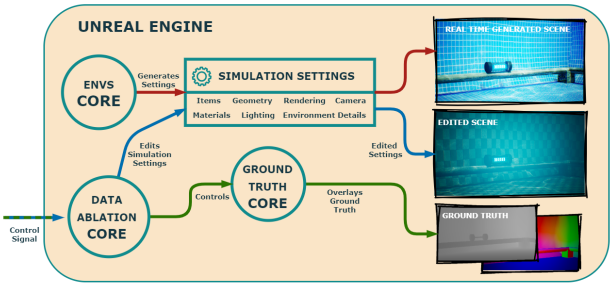

Figure 1: OpenWaters Simulation Architecture

Figure 1 depicts the architecture of our OpenWaters simulation toolkit. Below, we describe the key elements of the system.

Environments Core: This module contains the over-all code for generating novel scenes. It also contains several extendable abstract classes that can be used to target a wide range of domains. This covers the environment geometry and objects (e.g. pool or light transmitters), lighting (direct sunlight, diffused skylight, pool lights), water (clarity, color, surface level, surface agitation), rendering features(caustics, ray-tracing settings, resolution), Scene Materials and time (dilation or freezing).

Data Ablation Core: This module translates control signals from a human operator or a python script. These control signals apply changes in the environments generated by the environments core. We can adjust the cameras (camera matrix), lighting, Items, Rendering (e.g. reflection profiles or HDRI maps), Geometry and properties of the materials in the scene. Additionally, all of the ray-tracing engine parameters are exposed for experimentation within the Data Ablation module. The features of the data ablation core are adapted from [15, 16].

Ground-truth Core: OpenWaters includes scripts for automatically estimating depth, surface normals (world space and camera space), object masks and caustics segmentation. The Ground Truth Core is easily extendable by adding or modifying the existing scripts. We overlay these properties directly over the image with pixel-perfect alignment between the data and the ground-truth labels. In this work we explore only the depth ground truth core. This core annotates the calculated and normalized distance between each pixel that belongs to a specific object and the camera. By default, we set the real-life range of depth to 1000 centimeters, which covers the entire environment. This range is customizable in the GT Core.

4 EVALUATION

To *validate* our OpenWaters Simulation, we use it to generate a dataset of 13,000 640×480 RGB and Depth images, then we train a deep convolutional neural network (DCNN) from scratch to estimate depth in underwater scenarios. We chose depth since our real ZED camera did not produce reliable depth images underwater (See Figure 4.) Our images feature moderate to high levels of scattering and absorption (normal and murky water). In particular, since we aim to see the transference of learned features into the real domain for underwater depth estimation. Since reliable, accurate

ground-truth data is not available for underwater depth estimation in real domain, we exclusively use synthetic data for training. Our experiments show the model learns transferable features from the synthetic domain, that can be applied to real setup and infer depth from monocular images. Our data ablation experiments show the model attempting to extract perspective clues from the structure and texture, and using absorption and scattering as clues in predicting depth.

4.1 Experimental Setup

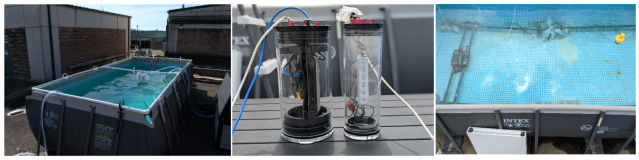

Figure 2: Our physical experimentation setup.

To gather real underwater images, we constructed a physical underwater setup. As Figure 2 shows, our physical setup is an Intex pool of size 20'×10'×5', with custom steel rails held together by cinder blocks that allow for horizontal movement of the camera and LED capsule. Our transmitter and the receiver (ZED stereo camera) [23] are submerged and mounted on the mentioned rails inside Blue robotics watertight enclosure capsules [5]. Our Simulation instance is modeled after this physical setup.

4.1.1 Simulation Instance. To model the physical setup, we create a set of custom 3D meshes. For the transmitter capsules, we model them inside Unreal Engine as per the technical details provided by their manufacturer [4]). We also model the Intex pool's dimensions in the physical setup. These meshes are available for public use with the released open-source code. The virtual transmitter and receiver are mounted similarly to our physical setup on virtual rails that allow for horizontal movement. The environments core generates a random location along the mounting bar to place the transmitter, and pseudo-random transform (location, pitch, yaw, roll) for the receiver. In each scenario, the transmitter and receiver are automatically mounted on random locations along the rails, and the objects are moved horizontally (similar to the physical setup) to introduce depth variety. Additionally, the main light source (sun) is randomly rotated in every frame to introduce lighting variety in the data.

Metrics. For evaluation, we used the same metrics as those used in [7]: average relative error (REL), root mean squared error (RMS), average log10 error, and threshold accuracy ($\delta_i < 1.25^i \; for \; i = [1, 2, 3]$).

Hardware. We conducted all our experiments on a Computer with an AMD Ryzen 5 5600x CPU, a GeForce RTX 2080 Ti graphics card, and 16 GBs of RAM.

Deep Neural Networks. We used the encoder-decoder architecture, and loss function from [17], with implementation from [2]. We train our model on purely synthetic data generated by OpenWaters simulation for 50 epochs. Dataset size is 13000 Images, split 80% (10400) for Training and 20% (2600) for validation.

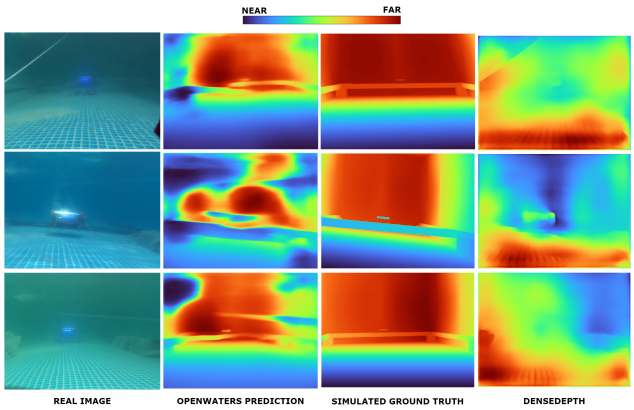

Figure 3: Qualitative Analysis - Performance of the model on Real images. From Left: Real image, Prediction of model trained on OpenWaters synthetic data, Ground-truth simulated from OpenWaters, Prediction of DenseDepth model.

5 RESULTS AND DISCUSSION

To *test* our model, we generate 80 images in various scenarios using the data ablation utilities in the simulation. For each scenario, camera and lighting are set at random positions, then the data ablation core freezes time to prevent changes in the caustics pattern, scattering and water particles. Then, it captures ground-truth and the test image. To accurately measure the effects of change in the isolated features, the test images are exactly the same, with only the specified parameters changing. (See Figure 5).

Performance on Real Domain. To get a better grasp of the model's performance, we recreated images from the physical setup in OpenWaters, where we have access to accurate depth ground-truth. These recreations are featured in figure 3. We then manually matched the real images and synthetic ground-truth by overlaying them pixel-by-pixel in Adobe Photoshop [1]. This allows us to run a *semi-quantitative* evaluation on the performance of the model. As seen in Table 3 performance of the model in real images is consistent with our data ablation diagnostic analysis on synthetic images. Overall, our model can generate depth maps in real underwater images without being trained on real data. Based on our qualitative and semi-quantitative analysis, it seems to have picked up useful features from its training with synthetic data. Figure 3 also shows predictions of our model vs. DenseDepth [2] in our real underwater setup.

Unreliable Depth Ground-truth in Real images. As seen in Figure 3, our model can generate a depth map by from a monocular real image without prior training on real data. We confirm the findings of other studies that depth cameras do not yield reliable results when submerged underwater [3]. Passive depth sensing cameras like the ZED Camera [23] use disparity matching to calculate a depth map. In underwater images, there are fewer structural key points to track for depth. Furthermore, lens-like curvature of the blue robotics capsule [5] used to submerge the camera setup underwater introduces an uneven pincushion distortion around the edges of the frame that further interferes with disparity matching in camera. As such, our ZED camera does not produce an accurate

Table 1: To evaluate the robustness of the model, we run a set of data Ablation tests in the same domain. As indicated by the results, Our model is robust to changes in lighting or murkiness. Arrows indicate if higher or lower values are better.

Experiment	Goal	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL↓	RMS↓	log10↓
Blue Intex Pool	Sanity Check (Same domain test set)	0.9699	0.9895	0.9949	0.0444	0.0214	0.0193
Murky	Effect of heavy scattering in same domain	0.9671	0.9885	0.9945	0.0486	0.0256	0.0216
No Sunlight	Effect of sunlight in same domain	0.9652	0.9877	0.9941	0.0529	0.0264	0.0234

Table 2: Data Ablation tests. In each experiment, image features are isolated to determine the result of experiment goal. Arrows indicate if higher or lower values are better

Experiment	Goal	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL↓	RMS↓	log10↓
Blue Intex Pool	Sanity Check	0.9699	0.9895	0.9949	0.0444	0.0214	0.0193
Procedural Tiles	Effect of change in pool texture	0.5878	0.6533	0.7198	0.1027	0.1255	0.2273
Procedural Tiles: Murky	Effect of heavy scattering + pool texture	0.6053	0.6698	0.7043	0.7968	0.1076	0.2723
Clay bottom	Effect of change in pool texture	0.5921	0.6781	0.7283	0.4378	0.1178	0.2567
Clay bottom: Murky	Effect of heavy scattering + pool texture	0.6007	0.6855	0.7408	0.6138	0.1423	0.2166
Ceramic Tiles	Effect of change in pool texture	0.3098	0.4458	0.5372	0.5971	0.2645	0.4428
Ceramic Tiles: Murky	Effect of heavy scattering + pool texture	0.5885	0.6437	0.6727	0.4417	0.1125	0.3465
Irregular Tiles	Effects of Non-Cubic texture	0.4184	0.5678	0.6734	0.5317	0.2171	0.2707
Irregular Tiles: Murky	Effects of Non-Cubic texture + Scattering	0.6540	0.7385	0.7887	0.4067	0.0847	0.1887
Dark Pattern	Effects of Non-Cubic texture	0.1641	0.3483	0.5089	0.6850	0.3116	0.3574
Dark Pattern: Murky	Effects of Non-Cubic texture + Scattering	0.6077	0.7201	0.7806	0.4193	0.0982	0.1909
Cobblestone	Effects of Non-Cubic texture	0.2794	0.4357	0.5108	0.5724	0.2603	0.5011
Cobblestone: Murky	Effects of Non-Cubic texture + Scattering	0.5590	0.6081	0.6288	0.4181	0.1226	0.3798

Table 3: Semi-Quantitative analysis of model performance on Real Images. Performance drop is consistent with our data ablation diagnostic results. These numbers serve as a rough representation of model performance on real images, and not its actual performance. Arrows indicate if higher or lower values are better.

Experiment	Goal	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL↓	RMS↓	log10↓
Blue Intex Pool	Sanity Check (Same domain test set)	0.9699	0.9895	0.9949	0.0444	0.0214	0.0193
Real Setup	Semi-quantitative analysis	0.3862	0.5709	0.7195	0.4215	0.2653	0.2556

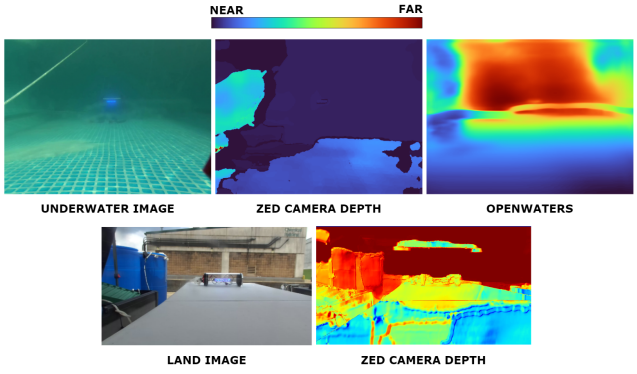

Figure 4: Comparison between ZED camera raw depth output and OpenWaters depth prediction. The ZED Camera output is unreliable in underwater scenarios.

depth map in the real underwater setup. As Seen in Figure 4, our depth estimation neural network can predict depth where the real camera produces an unreliable depth image.

5.1 Data Ablation Diagnostic Analysis

Table 1 shows the performance of our network, trained entirely on synthetic data generated by OpenWaters Simulation. In particular, the first table shows that our Depth estimation model achieves close to 97% ($\delta < 1.25^3$) accuracy in the test set, which contains challenging features such as water caustics, scattering and lighting variety. Our depth estimation model is quite robust to changes in lighting or scattering amount (murkiness). We argue this is due to these features being present in the training data. This table further confirms the feasibility of training deep learning models on synthetic data. In particular, it shows the model can learn to estimate depth from monocular synthetic images.

Table 2 shows the results for our data ablation tests (sample images in Figure 5) on the effects of pool texture and scattering on the performance of the depth estimation model. Below, we go over our insights from the Data Ablation diagnostic analysis.

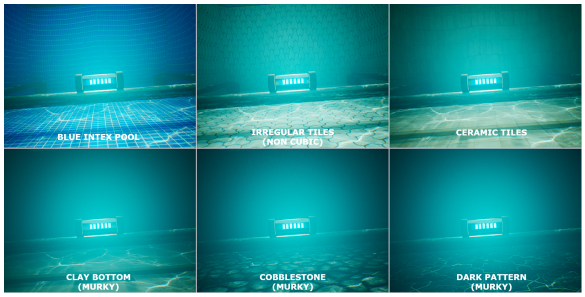

Figure 5: Data Ablation: Isolating specific features (in these images, murkiness and pool material) inside the exact same image, to evaluate their effects on the performance of a neural network. (Figure best viewed on screen)

Effect of change in pool texture. Using the data ablation core in OpenWaters, we perform experiments that are extremely difficult to do in real physical setups. One of which is introducing a change in the pool texture, while keeping all else exactly the same. The texture of the pool is directly related to the environment, since an object's surface material will affect its interactions with light, and it can dramatically change how it appears in an image. We see an expected performance drop similar to a dataset shift problem when switching to other types of texture materials for the pool.

Looking for Perspective clues. One reason for trying different pool textures is seeing the effect of perspective clues in the performance of the model. Many pools come with uniform, uni-color, often cubic tiles that can easily be tracked by the model as an easy perspective clue. In fact, this effect can be seen in our real images test, where the model guesses depth on the pool bottom almost perfectly. However, in case of a more complex texture like procedural tiles or non-cubic textures like cobblestone, we see a sharp decrease in performance. (see table 2) We argue this is because it is much harder for the model to extract perspective information from such irregular textures, compared to uniform tiles. We might be able to improve this by using the data ablation core to add more variety to the training data.

6 CONCLUSION & FUTURE WORK

In this paper, we introduced OpenWaters: an open-source, extendable and user friendly simulation kit that enables fine control over every variable, including realistic rendering parameters and ground-truth labels. We used OpenWaters to train a deep neural network, predicting depth from single underwater images. We demonstrated our model's ability to learn from synthetic images generated from OpenWaters, and provided semi-quantitative and qualitative analysis on how this model can perform in real images. In addition, we showcased the diagnostic tools provided with OpenWaters by performing data ablation diagnostic analysis on the depth estimation model to determine its weak points and infer insights for training neural networks for underwater computer vision. We posit that our analysis serves as an example of the types of experiments OpenWaters simulation kit enables in the future.

REFERENCES

[1] Adobe Inc. 2018. *PhotoShop CC19.1.2.* https://www.adobe.com/products/photoshop.html
[2] Lbraheem Alhashim and Peter Wonka. 2018. High Quality Monocular Depth Estimation via Transfer Learning. arXiv e-prints abs/1812.11941, Article arXiv:1812.11941 (2018). arXiv:1812.11941 https://arxiv.org/abs/1812.11941
[3] Atif Anwer, Syed Saad Azhar Ali, Amjad Khan, and Fabrice Mériaudeau. 2017. Underwater 3-D Scene Reconstruction Using Kinect v2 Based on Physical Models for Refraction and Time of Flight Correction. *IEEE Access* 5 (2017), 15960–15970. https://doi.org/10.1109/ACCESS.2017.2733003
[4] Blue Robotics. 2021. Technical Details of Watertight Enclosure for ROV/AUV (6" Series). https://bluerobotics.com/store/watertight-enclosures/6-series/wte6-asm-r1/#tab-technical-details
[5] Blue Robotics. 2021. Watertight Enclosure for ROV/AUV (6" Series). https://bluerobotics.com/store/watertight-enclosures/6-series/wte6-asm-r1/
[6] P. Drews, Jr., E. do Nascimento, F. Moraes, S. Botelho, and M. Campos. 2013. Transmission Estimation in Underwater Single Images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*.
[7] David Eigen, Christian Puhrsch, and Rob Fergus. 2014. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. *CoRR* abs/1406.2283 (2014). arXiv:1406.2283 http://arxiv.org/abs/1406.2283
[8] Epic Games. 2020. *Unreal Engine 4.26.* https://www.unrealengine.com
[9] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. 2016. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. arXiv:1605.06457 [cs.CV]
[10] Honey Gupta and Kaushik Mitra. 2019. Unsupervised Single Image Underwater Depth Estimation. arXiv:1905.10595 [cs.CV]
[11] Jie Li, Katherine A. Skinner, Ryan M. Eustice, and Matthew Johnson-Roberson. 2018. WaterGAN: Unsupervised Generative Network to Enable Real-Time Color Correction of Monocular Underwater Images. *IEEE Robotics and Automation Letters* 3, 1 (2018), 387–394. https://doi.org/10.1109/LRA.2017.2730363
[12] Na Li, Ziqiang Zheng, Shaoyong Zhang, Zhibin Yu, Haiyong Zheng, and Bing Zheng. 2018. The Synthesis of Unpaired Underwater Images Using a Multistyle Generative Adversarial Network. *IEEE Access* 6 (2018), 54241–54257. https://doi.org/10.1109/ACCESS.2018.2870854
[13] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. 2017. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. arXiv:1703.09312 [cs.RO]
[14] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. 2016. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*. 1–8. https://doi.org/10.1109/OCEANS.2016.7761080
[15] Mehdi Mousavi and Rolando Estrada. 2021. SuperCaustics: Real-time, open-source simulation of transparent objects for deep learning applications. arXiv:2107.11008 [cs.GR]
[16] Mehdi Mousavi, Aashis Khanal, and Rolando Estrada. 2020. AI Playground: Unreal Engine-based Data Ablation Tool for Deep Learning. In *International Symposium on Visual Computing*. Springer, 518–532.
[17] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*.
[18] NVIDIA. 2021. *NvRTX Branch of Unreal Engine.* https://github.com/NvRTX/UnrealEngine/tree/NvRTX_Caustics-4.26
[19] Yan-Tsung Peng, Xiangyun Zhao, and Pamela C. Cosman. 2015. Single underwater image enhancement using depth estimation based on blurriness. In *2015 IEEE International Conference on Image Processing (ICIP)*. 4952–4956. https://doi.org/10.1109/ICIP.2015.7351749
[20] Mario Prats, Javier Pérez, J. Javier Fernández, and Pedro J. Sanz. 2012. An open source tool for simulation and supervision of underwater intervention missions. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2577–2582. https://doi.org/10.1109/IROS.2012.6385788
[21] Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, and Steven M. Seitz. 2018. Soccer on Your Tabletop. *CoRR* abs/1806.00890 (2018). arXiv:1806.00890 http://arxiv.org/abs/1806.00890
[22] Shreeyak S. Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. 2019. ClearGrasp: 3D Shape Estimation of Transparent Objects for Manipulation. arXiv:1910.02550 [cs.CV]
[23] StereoLabs. 2021. ZED Stereo Camera. https://www.stereolabs.com/zed/
[24] Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. 2019. DIODE: A Dense Indoor and Outdoor DEpth Dataset. *CoRR* abs/1908.00463 (2019). http://arxiv.org/abs/1908.00463
[25] Yang Wang, Jing Zhang, Yang Cao, and Zengfu Wang. 2017. A deep CNN method for underwater image enhancement. In *2017 IEEE International Conference on Image Processing (ICIP)*. 1382–1386. https://doi.org/10.1109/ICIP.2017.8296508
[26] Olaya Álvarez Tuñón, Alberto Jardón, and Carlos Balaguer. 2019. Generation and Processing of Simulated Underwater Images for Infrastructure Visual Inspection with UUVs. *Sensors* 19, 24 (2019). https://doi.org/10.3390/s19245497