OysterNet: Enhanced Oyster Detection Using Simulation

Xiaomin Lin[1], Nitin J. Sanket[2], Nare Karapetyan[1], Yiannis Aloimonos[1]

***Abstract*—Oysters play a pivotal role in the bay living ecosystem and are considered the living filters for the ocean. In recent years, oyster reefs have undergone major devastation caused by commercial over-harvesting, requiring preservation to maintain ecological balance. The foundation of this preservation is to estimate the oyster density which requires accurate oyster detection. However, systems for accurate oyster detection require large datasets obtaining which is an expensive and labor-intensive task in underwater environments.**

To this end, we present a novel method to mathematically model oysters and render images of oysters in simulation to boost the detection performance with minimal real data. Utilizing our synthetic data along with real data for oyster detection, we obtain up to 35.1% boost in performance as compared to using only real data with our *OysterNet* network. We also improve the state-of-the-art by 12.7%. This shows that using underlying geometrical properties of objects can help to enhance recognition task accuracy on limited datasets successfully and we hope more researchers adopt such a strategy for hard-to-obtain datasets.

I. INTRODUCTION

Oyster reefs are filter feeders and they provide crucial benefits for the benthic marine ecosystem(s) such as increasingthe richness for a variety of species, providing living habitat, food, and protection for numerous marine species. However, the standing stocks for the oysters near the Chesapeake Bay [1] and North Sea [2] have dropped significantly due to over-fishing, global warming and the effect of diseases across the 19th century. To tackle this devastating ecological problem, massive efforts are being carried out to restore oyster habitats across the United States [3]–[6] and Europe [2].

One of the core challenges to advance, improve and adapt the restoration process is monitoring of the progress of oyster restorations effectively. Beck et al. [7] proposed to standardize monitoring metrics, units, and performance criteria for the evaluation of the oyster reefs. A set of environmental variables including water salinity, temperature, and dissolved oxygen are being monitored to determine the well-being of oyster habits. For the oyster reefs, universal parameters such as "reef areal dimensions, reef height, oyster density, and oyster size-frequency distribution" are monitored and reported in the literature [2], [4]–[7]. However, these metrics rely on recognizing and counting oysters, which is currently largely done by
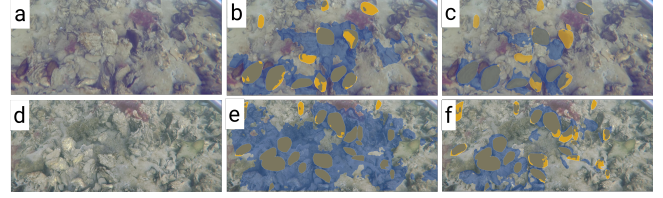
Fig. 1. Each row left to right: Input image, output of the network when trained using only real data, output of the network (which we call *OysterNet*) when trained using real data augmented with our synthetic data. Yellow represents the oyster segmentation ground truth and the blue is the predicted segmentation result. Notice how the number of false positives and false negatives drop significantly when the training data is augmented with our synthetic data, *All the images in this paper are best seen in color on a computer screen at 200% zoom.*

expert manual labor. Such a process is slow, time-consuming, and has poor scalability. Using such a manual approach, the oyster reefs can only be monitored within a restricted area with few samples, e.g., 100 oysters per sample site [5]. Furthermore, the material for the oyster's surface is similar to the seabed sediment (See Fig. 1a, Fig. 1d,). And it is significantly different from the washed oysters in Fig. 3a, which makes it very challenging to train new people or algorithms to perform oyster counting.

To streamline the process of oyster mapping, the goal is to utilize the advancements in robotics and artificial intelligence that can enable us to gather images from underwater Remotely Operated Vehicles (ROVs) and then automate the oyster detection and density calculation. The central part of this process is to build an oyster detection system. In this work, we present a mathematical model to generate oyster models and further use Generative Adversarial Networks to enable sim-to-real transfer. To the best of our knowledge, this is the first attempt to geometrically model oysters. The contributions of this paper are as follows:

- We propose a novel mathematical model for the 3D shape of oysters.
- We simplify the geometric model of an oyster for the projection on the image plane which is used to generate photorealistic synthetic oyster images. These images are used to train a deep segmentation network *OysterNet* for oysters that achieves the new state-of-the-art.
- We open-source our oyster generation model and dataset associated with this work to accelerate further research.

The rest of this paper is organized as follows: First, we place this work in the context of previous works in Sec. II. Then, we describe the proposed geometric model of the oyster which is used to create realistic images in Sec. III.We then present extensive quantitative and qualitative evaluations
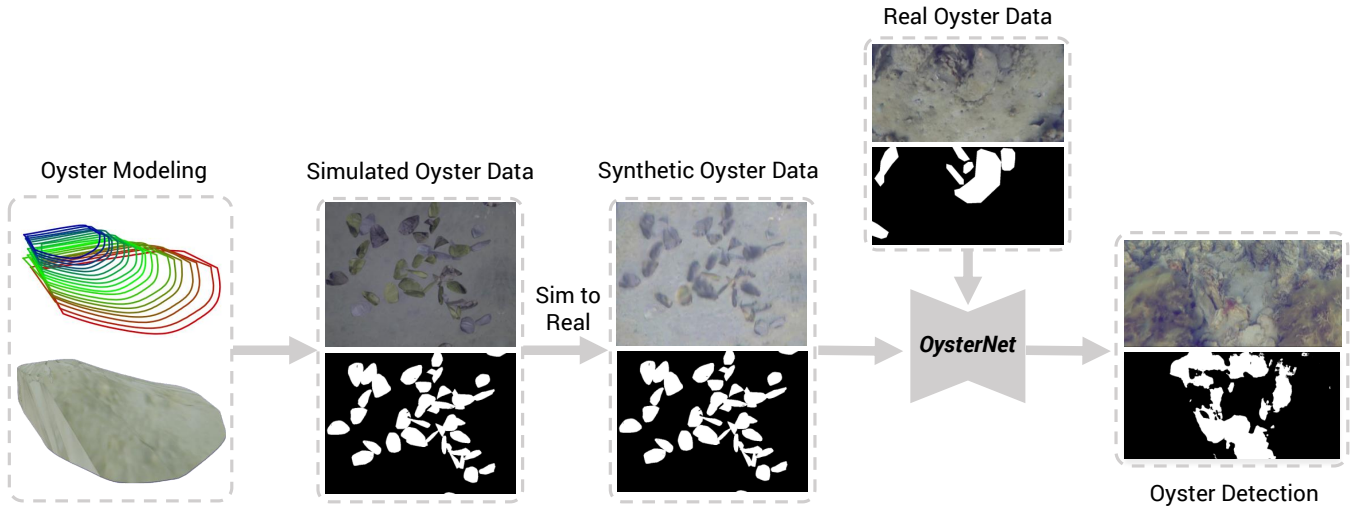
Fig. 2. An overview of our approach: The proposed geometric model is used to generate synthetic images which are further fed into a Generative Adversarial Network to enable sim-to-real transfer (domain adaptation) by generating photorealistic oyster images. We then combine the synthetic data with real data to train a *OysterNet* for oyster detection.

of our approach in Sec. IV. Finally, we conclude the paper in Sec. V with parting thoughts on future work.

II. RELATED WORK

Automation and robotics are becoming an integral part of many applications, particularly in the marine domain for environmental monitoring [8]–[10]. The marine domain poses many additional challenges on top of the classical ones faced by robotics. Some of the commonly encountered problems are image visibility distortions caused by the water or sediment, and the difficulty of acquiring data for developing recognition or planning methods that are driven by the information. This is especially true for oyster monitoring.

Systematic monitoring of underwater ecosystem requires reliable autonomous navigation. However, in underwater environments, navigation is challenging not only due to the dynamics of the systems but also due to a lack of adequate spatial awareness. To overcome these challenges vision-only based methods have been developed in the literature, that use human-labeled data to learn navigation commands for surveying coral reefs [11] and shipwrecks [12]. It is important to point out that these methods heavily depend on the quality of data and are hence require extensive data collection for a high degree of robustness.

Sadrfaridpour et al. [13] recently collected an underwater dataset and used Mask-R-CNN [14] for oyster detection. However, this dataset is relatively small and is not collected in the ocean/sea bed but rather on an oyster farm. The oysters are stacked and are very dense in the dataset which lead to poor detection results for oyster reefs in the real world. There is a lack of variety of oysters and environment in Sadrfaridpour's dataset for a more robust detection result when deployed in the wild.

Instead of collecting large datasets for detection tasks, we follow the conceptual approach proposed by Sanket et al. [15] which is to geometrically model the object under

consideration to generate an enormous amounts of data synthetically. In particular, we model the 3D shape of oysters to create an underwater dataset for oyster detection.

Alternatively, Generative Adversarial Networks (GANs) has been also widely used to generate underwater datasets. Joshi et al. [16] utilized a GAN to create images for pose estimation of an autonomous underwater vehicle (AUV). However, this method also required a large number of ground truth data samples for generating all possible pose images. To tackle the visibility distortions caused by the water or sediment, Li et al. [17] proposed a system that restores underwater images into in-air images. Moreover, Wang [18] also proposed an approach for real-world underwater color restoration and dehazing by utilizing GANs.

With the ultimate goal of developing an autonomous information-driven oyster monitoring system, we acknowledge the need of good oyster prediction methods. Moreover due to the lack of large samples of oyster detests and the lack of literature to address this problem we are seeking an alternative path for generating a large oyster dataset by looking into the mathematical model of oysters. To the best of our knowledge, we are the first to propose a geometric model of oysters and use that model to generate synthetic data which is described next.

III. SYNTHETIC OYSTER GENERATION

In this work, we propose a novel method for modeling oysters which are then used to generate a large dataset of oyster reef images. In this section, we will talk about our proposed method (which is summarised in Fig. 2). First, we will describe the process of modeling oysters. Next, we will talk about how this model can be used to generate photorealistic images of oyster reefs using domain adaptation. Last but not least, we train a semantic segmentation model using the *OysterNet* for detecting the oysters in the real world. We present each sub-part in the following sub-sections.

Fig. 3. Steps in the geometric modelling of an oyster: (a) Sample image of a real oyster shell, (b) 3D scan of a real oyster, (c) Splines fit to the oyster's bottom layer to model it (each color represents a single spline), (d) Simplified model of one stratified layer of the oyster (single layer of spline curve $S(t)$), (e) all layers of $S^\alpha(t)$, (f) generated 3D model of oyster, (g) final generated 3D model of oyster with added real oyster texture.

A. Geometric Modelling Of Oysters

In order to model the geometry of an oyster, we first 3D scanned ten washed oysters (sample image example shown in 3a and scanned model shown in 3b) which were used to build a mathematical model. Furthermore, the parameters in the proposed mathematical model can be adjusted to generate models of various oysters. We will describe the mathematical model next.

It is important to note that the geometrical shape of an oyster is extremely complex, with each one having a different shape and thickness. Oysters grow following a general oyster shape but expand somewhat randomly along their margins. In the first step, we will model our oysters in 3D. Each oyster can be approximated as a series of mathematical functions in a stratified manner (akin to the way a 3D printer prints). We start with each 'horizontal' layer (slice of the cross-section of the oyster) by looking at the top view of our scanner oysters. We call this the perimeter of the oyster. We noticed that the perimeter can be easily modeled using two cubic B-splines. Let the $n+1$ control points for the splines be $\{c_0, ..., c_n\}$ and $m+1$ knot vectors be $\{t_0, ..., t_m\}$, then the spline curve $S(t)$ of degree k is given by

$$S(t) = \sum_{i=0}^{n} c_i B_{i,k}(t) = 1, \tag{1}$$

where, $B_{i,k}(t)$ denotes the basic function of degree k and is computed recursively as

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t_{i+1} \geq t \geq t_i \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} B_{i,k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(t). \tag{3}$$

In our case here, $m = n + k + 1$ and we selected $k = 3$ for a cubic spline. Particularly, we utilize two cubic B-splines: one for the top half of the shell, and one for the bottom of the shell (See Figs. 3c and 3d). We will call this perimeter model (Eqs. 1, 2 and 3) as the 2D model since it models only a single layer of the oyster.

Now, we want to extend the 2D model to 3D in the stratified manner we described before. However, having a high resolution in depth is computationally prohibitive due to all the nooks and crannies (high-frequency edges) on the oyster shell (See Fig. 3b). In order to make this computation tractable, we simplify the shape of the oyster by assuming that it is smooth since the variation on the shell is much

smaller than the distance to the oyster. It is important to highlight that the visual changes these high-frequency edges created are approximated by the visual textures we place on our generated oysters. The 3D model of the oyster (Fig. 3e) follows the same perimeter model from before but also adds changes to depth. Let $c_0^\alpha, ..., c_n^\alpha$ denote the control points for α-th layer of the 3D oyster. And the knot vectors for the α-th layer be $\{t_0^\alpha, ..., t_m^\alpha\}$. We define the c_n^α, t_m^α as follows

$$c_n^\alpha = c_n + X_1 \tag{4}$$

$$t_m^\alpha = t_m + X_2 \tag{5}$$

where $X1$ and $X2$ are the Gaussian Noise used to model the high frequency edges of the oyster's perimeter. Formally, $X1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$.

We can then substitute c_n, t_m with c_n^α, t_m^α in Eqs. 1 to 3 to get the spline curve $S^\alpha(t)$ for every single layer.

For each spline curve $S^\alpha(t)$, we can use $[x'_\alpha, y'_\alpha]^T$ to represent spline curve points as follows:

$$[x_\alpha, y_\alpha]^T = [x'_\alpha, y'_\alpha]^T r^\alpha \tag{6}$$

where r is the in-growth rate as a fixed number.

Finally, the growth rate of the oyster is defined as follows:

$$z_\alpha = \alpha d, \tag{7}$$

where d is the depth at a fixed number here.

Now we can use all the α layers with $[x_\alpha, y_\alpha, z_\alpha]^T$ points to generate 3D model (Fig. 3f) by using pyvista [19] data visualizer. By varying the parameters $\Theta = \{\sigma_1, \sigma_2, \mu_1, \mu_2, \alpha\}$ we obtain different oyster shapes (Fig. 7). Further, we use image textures from real oysters collected in the Chesapeake Bay to be warped on the generated 3D model (See Fig. 3g).

In the next section, we will talk about how the actual images are rendered from the 3D models we just constructed.

B. Synthetic Image Generation

1) Simulation Image Rendering: A 3D model is not sufficient for creating images for oyster segmentation. We utilized the Blender™ [20] game engine to simulate the oysters on a seabed. We rendered 13K synthetic oysters with different 3D models (we will just call it the synthetic model for simplicity) by varying parameters Θ as described in Sec. III. Then we used these synthetic 3D models for synthetic data generation. First, we employed 14 real oyster texture images and applied them randomly to all the synthetic models (Fig. 4a). A sample of the synthetic model with

applied texture image can be seen in Fig. 4b. To generate an simulated oyster reef image, we placed the oysters with random poses onto a flat surface with the seabed textures as shown in Fig 4c. Images of the oyster reef are then rendered in Blender^TM along with segmentation masks (Fig. 4d).

However, Fig. 4c is not photorealistic and does not look like a real underwater image which will lead to a poor detection performance when deployed. We describe how we use a Generative Adversarial Network (GAN) to perform sim-to-real domain adaptation such that our images are photorealistic such that they can generalize to the real world after being trained in simulation.

2) Domain Adaptation: To render a realistic oyster image, we employ contrastive unpaired translation (CUT) [21] for unpaired image-to-image translation by learning functions to map from the Blender^TM [20] domain R to the target domain T. The overall loss function consists of three parts: GAN loss, PatchNCE loss, and ExternalNCE loss which are described next.

GAN Loss: Generators G and F are used to transfer domains: $G : R \rightarrow T$ and $F : T \rightarrow R$. In order to
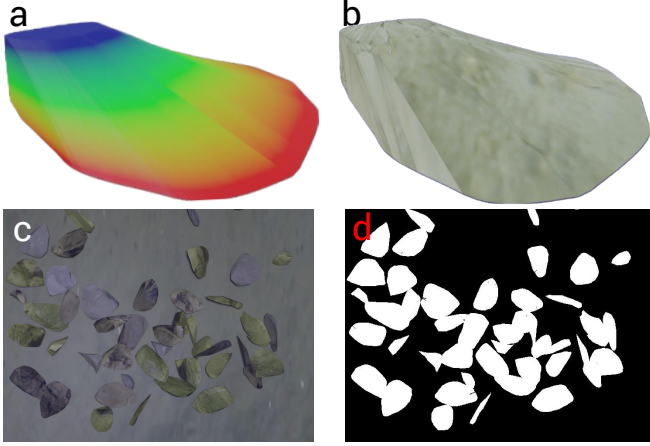

Fig. 4. (a) One of the synthetic models generated from Sec. III-A), (b) Synthetic model with real oyster texture added, (c) image of an oyster farm with 50 synthetic oysters generated in Blender^TM, (d) masks for oysters in (c).
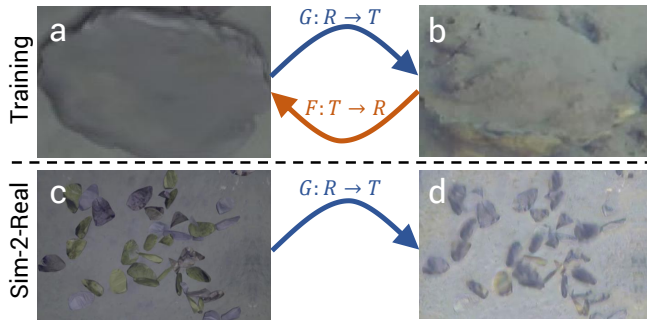

Fig. 5. Domain Adaptation to perform sim-to-real transfer. (a) A single sample of the simulated oyster, (b) A single sample of the real oyster, (c) Simulated oyster farm, (d) Synthetic oyster farm domain adapted to real world for photorealism.

improve image-to-image translation in CUT, ensuring the reconstructed images $F(G(R)) \approx R$ is necessary. Thus, we want to minimize the adversarial loss. Adversarial loss is defined by the following two components: discriminator loss and generator loss. Where discriminator loss is to minimize the loss from misclassification between real and fake samples. As for the generator loss, the goal is to maximize the discriminator's probabilities of being real.

PatchNCE loss: We first break the function G into two parts, one encoder, and one decoder. $\hat{T} = G(R) = G_{dec}(G_{enc}(R))$. G_{enc} is used for image translation. Therefore, a patch of the input images can be represented as the feature stack of each layer, and the spatial location of the feature stack from the encoder G_{enc}. We want to ensure the cross-entropy between these feature stacks from different layers and spatial locations is minimized. Then the PatchNCE loss is introduced where NCE represents Noise Contrastive Estimation.

ExternalNCE loss: Not only do we want to minimize PatchNCE loss, but we can also use the image patches from the rest of the dataset while training. A random negative image from the dataset is encoded and is used to define externalNCE loss.

We refer the readers to [21] for a detailed description of the Loss function and network. In our proposed system, the target domain is the realistic environment T_{real}.

3) Details of Training the CUT: We want the CUT network to be able to capture the synthetic oyster to real oyster translation. So, we extract 8959 images of single oysters from the synthetic data that we generated as R and we also extract 957 samples of single oysters from the real underwater images as T_{real}. In the training phase, we learn two mapping functions: $G : R \rightarrow T$ and $F : T \rightarrow R$. Once the CUT is learned, we perform inference on our synthetic images to make them look photorealistic (See Fig. 5) by performing domain transfer which is further used to train our oyster detection network *OysterNet*. In this work, we train CUT for 153 epochs to obtain our generator G.

IV. EXPERIMENTS AND RESULTS

First, we describe the datasets used in this section. To validate our generated oyster model, we then compare the results obtained by two different segmentation networks and two different datasets. Lastly, we experiment with how different values of the oyster model parameters affect the segmentation results.

A. Description of Datasets

Rendered/Synthetic Dataset: contains images obtained by randomly rendering generated 3D oyster models on the seabed using Blender^TM. Rather than just laying the oysters on the seabed, we simulate the randomization pose of the oysters in various realistic positions. This randomness in oyster pose makes the neural network more robust for recognizing the oyster in different poses. After the images
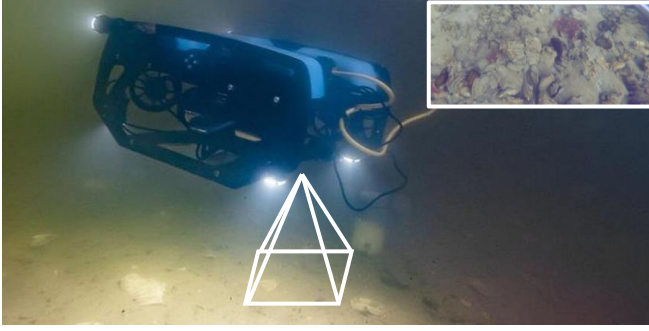
Fig. 6. BlueROV collecting data for our real dataset. The inset shows a sample image captured which is a part of our real dataset.

are generated from Blender™, we used CUT, as described in Sec. III-B, to create photo-realistic images. The rendered dataset contains 4800 images.

Real Dataset: contains images from Harris Creek taken by Chesapeake Bay Foundation by driving a BlueROV as shown in Fig. 6. We labeled the images by hand. We have 29925 images of size 256×256. There are over 3900 oysters labeled in the dataset. Unlike some other datasets for underwater object detection, oysters are really hard to recognize, even for humans. So, we worked with experts at the Chesapeake Bay Foundation to obtain this dataset.

B. Experimental Results

OysterNet adapts UNet [22] as its backbone. We trained *OysterNet* with a learning rate of 0.001 with decay. We use Adam optimizer coupled to the Jaccard loss [23] loss function. We used a batch size is 32 and trained the network for 100 epochs.

We want to verify our hypothesis that the synthetic oyster data we generated can help improve the oyster semantic segmentation.

[13] obtained the best results for oyster detection using the FPN [24] network. This is the model we use which we denote as the DCO (Detecting and Counting Oysters) method. In our observation UNet [22] architecture performed better using our approach which refer to it as *OysterNet* ("Ours") in Table I.

We want to use the minimal amount of real data for training as they are hard to obtain, expensive, and labor intensive. To this end, we use only 25% of the real dataset we used as training which we will denote as O_{real}. We test our method on the remaining 75% of the real dataset that we call O in the Test Data. We denote all the images in our generated synthetic dataset as O_{syn} which will be used for training with/without real data. Formally, $O_{syn\_and\_real}$ is the combination of O_{real} and O_{syn}.

In this experiment, we perform training on O_{real} (our real dataset) and test it on the O (our held-out real dataset) with both *OysterNet* and DCO methods. The Intersection over Union (IoU) scores are 18.16% and 18.88% respectively which serves as the baseline for oyster segmentation results for our dataset. Both the methods perform similarly in this

TABLE I

COMPARISON OF SEMANTIC SEGMENTATION RESULTS WITH THE STATE-OF-THE-ART.

Method	Train Data	Test Data	IoU Score(%)
DCO [13]	O_{syn}	O	6.47
DCO [13]	O_{real}	O	18.88
DCO [13]	$O_{syn\_and\_real}$	O	21.76
OysterNet (Ours)	O_{syn}	O	7.45
OysterNet (Ours)	O_{real}	O	18.16
OysterNet (Ours)	$O_{syn\_and\_real}$	O	**24.54**

case. Then we want to observe the results when trained only on the generated synthetic dataset (O_{syn}). We train on O_{syn} and test on O with both methods. The IoU score is 7.45% and 6.47% respectively which is lower than our baseline. The network has learned to recognize oysters in the synthetic domain but the sim-to-real domain transfer is still not as desired. To this end, we add a small amount of real data to the synthetic data and use it for training ($O_{syn\_and\_real}$). We achieve a state-of-the-art IoU Score of **24.54%** against the expert human labeled ground truth which is 35.1% better than just using real dataset for training and 12.7% better than DCO when trained on synthetic augmented real data.

The results are tabulated in Table I. It shows that with the added synthetic dataset, we achieve a significant $\sim$35.1% accuracy improvement over just using the real dataset for training. As we can see from Figs. 1b and 1e, the network predicts mostly sediments as oysters when trained using only real data. This leads to a lot of false positives and false negatives. However, when the network is trained using real data augmented with our synthetic data (which we call *OysterNet*), there is a significant decrease in false predictions as we can see from Figs. 1c and 1f.

Since the generated dataset affects accuracy significantly, we ablate on how different parameters in our proposed geometric model affect performance in the next section.

C. Ablation Studies

We varied the parameters Θ that controlled amount of high-frequency components and the number of layers to generate oyster models. We also varied the size of the oyster when rendering it in the simulation. When not specified, the parameters are set as $\mu_1 = 150$, $\mu_2 = 150$, $\sigma_1 = 150$, $\sigma_2 = 15$, $\alpha \in [15, 20]$, and $Scale \in [25, 30]$. Only one parameter is varied at a time while keeping others fixed to the values specified. No two oysters are the same with respect to size and height, therefore we chose different ranges for α and Scale. We see from Table IIa that IoU Score increases with mean of the noise and then decreases.

This is because the noise increases, and the variety of the oysters increases which would benefit the detection of the oysters. However, when the noise is too large ($\mu_1 > 150$), the oyster generated no longer looks like an oyster (See Fig. 7). The IoU Score drops significantly when $\mu_1 = 250$. A similar trend is observed with μ_2 and σ_1 with the IoU Score peaking around 150 (Table IIb and IIc). We also observe that

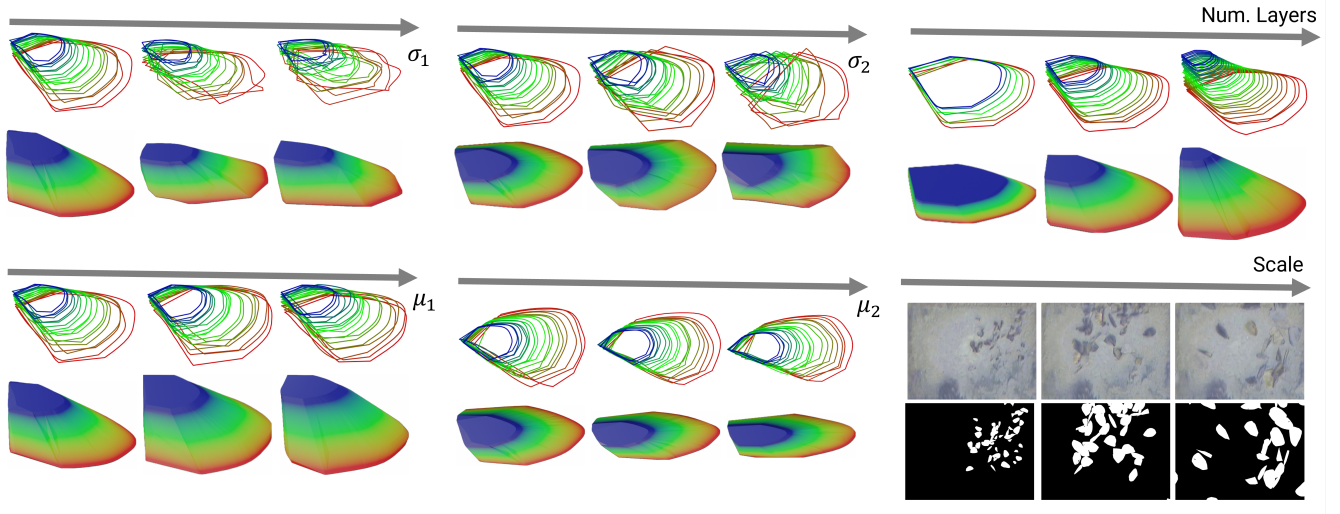| Scenario a						Scenario b				
μ_1	50	100	150	200	250	μ_2	50	100	150	200	250
IoU(%)	21.70	23.10	24.54	23.21	17.67	**IoU(%)**	24.19	21.23	24.54	22.34	19.57
Scenario c						Scenario d				
σ_1	50	100	150	200	250	σ_2	15	30	60	120	240
IoU(%)	19.37	22.07	24.54	23.64	20.67	**IoU(%)**	24.54	21.53	23.64	22.42	22.95
Scenario e						Scenario f				
α	5-10	10-15	15-20	20-25	25-30	**Scale(%)**	10-15	15-20	20-25	25-30	30-35
IoU(%)	22.76	22.88	24.54	22.63	22.66	**IoU(%)**	18.97	23.21	24.36	24.54	21.20

Fig. 7. Qualitative demonstration of how different parameters of our oyster model affect the shape of the oyster. The parameters are in the same order as the experiments in Table II.

with increase in standard deviation (σ_2) for X_2 (Table IId), the detection accuracy drop slightly. By varying the number of layers (α) for the oysters, we notice that the change in the IoU Score is relatively minimal. Since the image that we generated is in 2D, the parameter for layers (α) does not have a significant effect (Table IIe) as we mostly observe oysters from far away and in mostly top view.

In Table IIf, we observe that the scale of the oyster has to match the size of the oyster in the image to get the optimal result. The accuracy drops when the scale of the oyster is either too large or too small. The IoU Score difference between the scale values in the ranges of $20-25$ and $25-30$ is only 0.18% (Tables IIf).

V. CONCLUSIONS

In this work, we first model the geometry of oyster shells and render the oyster images in a game engine. Then we perform an image-to-image transformation from the simulation domain to the real-world domain. With the help of the generated synthetic dataset, when augmented to the real dataset we showed an improvement in the semantic segmentation IoU score for the oysters by 35.1% over just using real data for training and 12.7% over the current state-of-the-art. These results highlight that for data-critical applications when collecting real images is challenging,

it is possible to model the images using the underlying geometry of the object to create photorealistic images that will improve object detection drastically. To the best of our knowledge, this is the first attempt to model 3D structure of oysters. Being the first work in this field, there are many directions and possible improvements that can be made to our *OysterNet* framework to increase the accuracy of semantic segmentation. One possible improvement will be to model both shells of the oysters instead of a single shell. As for the detection phase, new network architectures can be explored to tackle this problem. Finally, more models of the oysters along with additional shape noise could be utilized in the future for a more robust detection.

ACKNOWLEDGMENT

REFERENCES

[1] R. I. Newell, "Ecological changes in chesapeake bay: are they the result of overharvesting the american oyster, crassostrea virginica," *Understanding the estuary: advances in Chesapeake Bay research*, vol. 129, pp. 536–546, 1988.

[2] B. Pogoda, "Current status of european oyster decline and restoration in germany," *Humanities*, vol. 8, no. 1, p. 9, 2019.

[3] L. P. Baggett, S. P. Powers, R. D. Brumbaugh, L. D. Coen, B. M. DeAngelis, J. K. Greene, B. T. Hancock, S. M. Morlock, B. L. Allen, D. L. Breitburg, *et al.*, "Guidelines for evaluating performance of oyster habitat restoration," *Restoration Ecology*, vol. 23, no. 6, pp. 737–745, 2015.

[4] B. N. Blomberg, T. A. Palmer, P. A. Montagna, and J. B. Pollack, "Habitat assessment of a restored oyster reef in south texas," *Ecological Engineering*, vol. 122, pp. 48–61, 2018.

[5] K. McFarland and M. P. Hare, "Restoring oysters to urban estuaries: Redefining habitat quality for eastern oyster performance near new york city," *PloS one*, vol. 13, no. 11, p. e0207368, 2018.

[6] S. J. Theuerkauf, D. B. Eggleston, and B. J. Puckett, "Integrating ecosystem services considerations within a gis-based habitat suitability index for oyster restoration," *PloS one*, vol. 14, no. 1, p. e0210936, 2019.

[7] M. W. Beck, R. D. Brumbaugh, L. Airoldi, A. Carranza, L. D. Coen, C. Crawford, O. Defeo, G. J. Edgar, B. Hancock, M. C. Kay, *et al.*, "Oyster reefs at risk and recommendations for conservation, restoration, and management," *Bioscience*, vol. 61, no. 2, pp. 107–116, 2011.

[8] S. Manjanna, N. Kakodkar, M. Meghjani, and G. Dudek, "Efficient terrain driven coral coverage using gaussian processes for mosaic synthesis," in *2016 13th Conference on Computer and Robot Vision (CRV)*. IEEE, 2016, pp. 448–455.

[9] J. Hansen, S. Manjanna, A. Q. Li, I. Rekleitis, and G. Dudek, "Autonomous marine sampling enhanced by strategically deployed drifters in marine flow fields," in *OCEANS 2018 MTS/IEEE Charleston*. IEEE, 2018, pp. 1–7.

[10] N. Karapetyan, J. Moulton, and I. Rekleitis, "Meander-based river coverage by an autonomous surface vehicle," in *Field and Service Robotics*. Springer, 2021, pp. 353–364.

[11] T. Manderson, J. C. G. Higuera, R. Cheng, and G. Dudek, "Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1885–1891.

[12] N. Karapetyan, J. V. Johnson, and I. Rekleitis, "Human diver-inspired visual navigation: Towards coverage path planning of shipwrecks," *Marine Technology Society Journal*, vol. 55, no. 4, pp. 24–32, 2021.

[13] B. Sadrfaridpour, Y. Aloimonos, M. Yu, Y. Tao, and D. Webster, "Detecting and counting oysters," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2156–2162.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[15] N. J. Sanket, C. D. Singh, C. M. Parameshwara, C. Fermüller, G. C. de Croon, and Y. Aloimonos, "Evpropnet: Detecting drones by finding propellers for mid-air landing and following," *arXiv preprint arXiv:2106.15045*, 2021.

[16] B. Joshi, M. Modassir, T. Manderson, H. Damron, M. Xanthidis, A. Q. Li, I. Rekleitis, and G. Dudek, "Deepurl: Deep pose estimation framework for underwater relative localization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1777–1784.

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

[18] N. Wang, Y. Zhou, F. Han, H. Zhu, and J. Yao, "Uwgan: underwater gan for real-world underwater color restoration and dehazing," *arXiv preprint arXiv:1912.10269*, 2019.

[19] C. Sullivan and A. Kaszynski, "Pyvista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk)," *Journal of Open Source Software*, vol. 4, no. 37, p. 1450, 2019.

[20] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[21] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *European Conference on Computer Vision*. Springer, 2020, pp. 319–345.

[22] 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*. Springer, 2015, pp. 234–241.

[23] J. Bertels, T. Eelbode, M. Berman, D. Vandermeulen, F. Maes, R. Bisschops, and M. B. Blaschko, "Optimizing the dice score and jaccard index for medical image segmentation: Theory and practice," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2019, pp. 92–100.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.