Comparing Photorealism in Game Engines for Synthetic Maritime Computer Vision Datasets

Kashish Sharma*, Borja Carrillo-Perez** and Sarah Barnes**

***Abstract*— Computer vision for real-world applications faces data acquisition challenges, including accessibility, high costs, difficulty in obtaining diversity in scenarios or environmental conditions. Synthetic data usage has surged as a solution to these obstacles. Leveraging game engines for synthetic dataset creation effectively enriches training datasets with increased diversity and richness. The choice of the game engine, pivotal for generating photorealistic simulations, may influence synthetic data quality. This study compares Unity Engine's and Unreal Engine's capabilities in generating synthetic maritime datasets to support ship recognition applications. To this end, the real-world maritime dataset ShipSG has been replicated in the corresponding game engines to create the same scenarios. The performance of the generated synthetic datasets is benchmarked against the real-world ShipSG dataset using the object recognition model YOLOv8. Furthermore, the comparison evaluates various photorealistic parameters found in the dataset images to determine the optimal configuration for improving performance with YOLOv8. The datasets generated using the Unity Engine, with all photorealistic effects present and the one with no lens distortion, achieved the highest accuracy in ship recognition with a mAP of 72.3%. Both configurations of the synthetic datasets were utilised to augment the ShipSG dataset to train YOLOv8. The configuration with all photorealistic parameters in place provides the highest mAP increase, of 0.4% compared with YOLOv8 performance on ShipSG when no synthetic data is used. This evidence underscores that utilising game engines can effectively support and enhance ship recognition tasks.**

***Index Terms*—Synthetic Data Generation, Game Engines, Photorealism, Maritime Computer Vision, YOLOv8**

I. Introduction

The field of computer vision has witnessed significant advancements, driven by the availability of large-scale datasets. Ideally, these datasets contain images that capture real-world events and occurrences. However, acquiring images of events in action is not always possible. Factors like the cost of data acquisition, along with privacy and ethical considerations, hinder dataset creation. In the maritime domain, the logistical and environmental challenges add to the problems associated with data acquisition. [1]

Synthetic datasets have emerged as a promising alternative, wherein the limitations associated with data acquisition can be overcome [2]. Synthetic datasets offer the ability to generate datasets on demand, allowing scalability, reproducibility and customisation [3]. However, before utilising such datasets in computer vision tasks, selecting the right tool for generation is crucial. Earlier, the synthetic datasets in computer vision were generated by crafting or staging simulated real scenarios [4] [5]. Recently, game engines have emerged as the favoured platform for generating and simulating real-world scenarios [6] [7]. However, the rationale or criteria for selecting a specific game engine for synthetic data generation in computer vision applications has not been defined in the literature. The works that specified their choice, called out their familiarity or the ease of use with that engine [8] [9] , the choice of the game engine has not been qualitatively analysed in the literature.

This work quantitatively compares the efficacy of two popular game engines[1]; Unity Engine [10] and Unreal Engine [11], in generating photorealistic synthetic datasets for maritime ship segmentation. The maritime computer vision dataset enhances situational awareness by enabling information extraction for infrastructure protection and implementing measures to address threats [12]. For a better understanding of the effects of photorealism in synthetic data that most impact our maritime computer vision task of ship segmentation, various simulation parameters are varied, and the resulting model precision is compared. We evaluate each synthetic dataset by analysing the mean Average Precision (mAP) achieved by an object detector trained on real images. Subsequently, we select the configuration with the highest mAP to augment model training data to observe the impact of the addition of synthetic data to the training dataset. This investigation aims to guide future synthetic dataset creation efforts, and lay down the foundation for the use of game engines as a tool for creating synthetic datasets, ensuring an optimised training environment for computer vision applications in the maritime domain and beyond.

II. Synthetic Data Generation and Game Engines

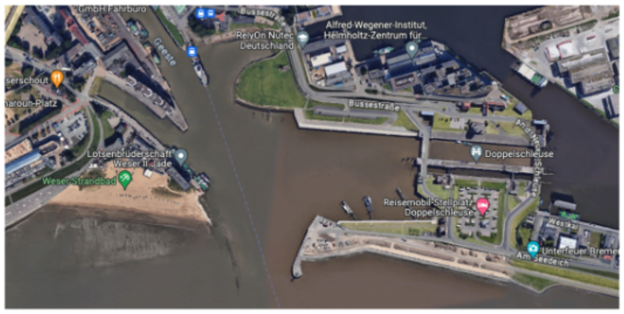
In Section II-A, we present the maritime dataset selected for this study. Subsequently, in Section II-B, we detail the process of constructing a true-to-scale model of the real-world location, which will be used for generating synthetic datasets discussed in Section II-C.

A. The ShipSG Dataset

The maritime domain requires images from real-world that showcases varied ship data with precise annotations including the ship class, masks and features like geographical coordinates. Existing datasets like, Singapore Maritime Dataset [13], SeaShips [14] suffer from a lack of annotations for instance

*University of Bremen, Germany
**German Aerospace Center (DLR), Institute for the Protection of Maritime Infrastructures, Bremerhaven, Germany.

[1]Link:https://www.slashdata.co/post/did-you-know-that-60-of-game-developers-use-game-engines, accessed on 22.07.2024


(a) Google Maps - Top View [15]

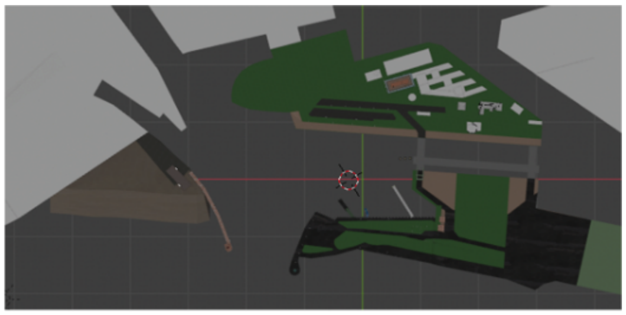

(b) Doppelschleuse Model - Top View in Blender

Fig. 1: Top View of the Doppelschleuse

segmentation, limited diversity in ship types, and absence of a favourable oblique view, all of which complicate the process of geo-referencing. To overcome these obstacles, the ShipSG dataset [12] for ship segmentation and geo-referencing available in the public domain was created. The images in the dataset (Fig. 3a) showcase the Doppelschleuse, port basin in Bremerhaven, Germany. The dataset consists of 3505 images with static and oblique views of the port basin and manually annotated 11,625 ship masks grouped into seven classes. The seven classes are Cargo, Law Enforcement, Passenger/Pleasure, Special 1, Special 2, Tanker and Tug. The dataset is split into two sets - training and validation. The training set contains 80% of the images, while the validation set contains the remaining 20%.

B. 3D Model Creation

To recreate the scenarios from the dataset in the game engines, a true-to-scale model of the Doppelschleuse (Fig. 1b) was handcrafted in the 3D modelling software Blender[2]. To ensure that the 3D model of the Doppelschleuse is true to scale, Google Maps [15], along with its measure tool, was employed to verify the model's adherence to real-world proportions (Fig. 1). The area under consideration measures 1000m in length and 500m in width.

The images (Fig. 2a) in the ShipSG dataset were captured from the rooftop of the Alfred Wegener Institute (AWI) at the Doppelschleuse, Bremerhaven, Germany. The end of the rooftop lies in the foreground (Fig. 2b) of the images. In the mid-ground (Fig. 2b) lies the street, the Molenfeuer and the Nordmole lighthouses along with the Lotsenstation. In the background (Fig. 2b), we have distant Bremerhaven to the right and the water body. Majority of vessel activity in the ShipSG dataset is concentrated in the foreground and mid-ground. Therefore, the land and the structures in the background are omitted in the 3D model (Fig.2c).

The structures in the foreground and the mid-ground are chosen for 3D modelling. Google Maps is used to trace the position and orientation of the structures. Particular care was taken to ensure that these structures displayed a high level of detail.

To recreate the vessels in the ShipSG dataset, we acquired 3D models of 21 vessels from Turbosquid[3] that closely resemble those in the real dataset. The vessel models were then scaled to match the dimensions of their real-world counterparts.

C. Synthetic Dataset Generation

In order to make sure that the comparison is fair for both the game engines, a baseline criteria was set. Both game engines should have similar rendering capabilities, using the most recent stable versions and to utilise the add-ons that are provided only by the developers. Based on this, Unity Engine HDRP 2023.2 and Unreal Engine 5.3 are selected for comparison.

To ensure real scenarios are replicated, we selected 52 images from the validation set of the ShipSG dataset, for reconstruction using game engines. These images depict various scenarios, including diverse times of day, lighting and foggy conditions, and different vessel types. The 3D Doppelschleuse model and the vessel models were imported into both game engines.

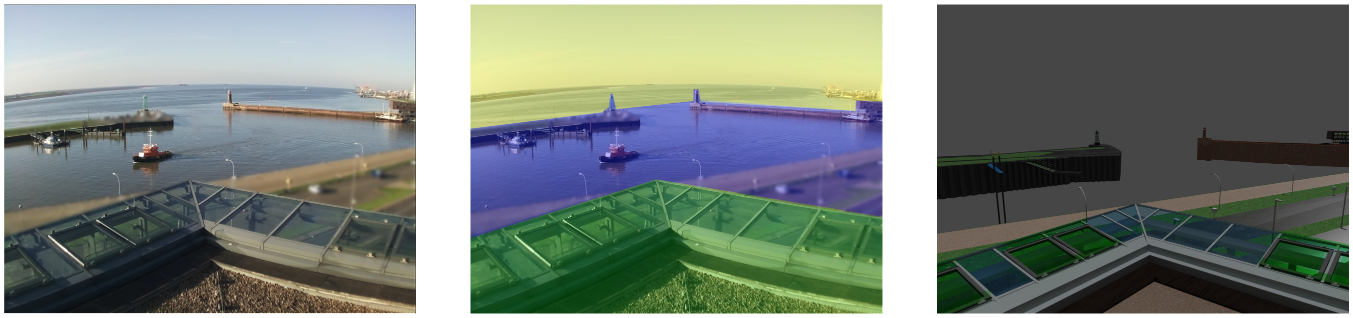
To analyse the significance of the photorealistic effects that contribute in making synthetic images appear photoreal, six effects were identified from the real images. The effects are:

1) All effects present (sun light, water body, lens distortion and colours) (Fig. 7a in Appendix)
2) Absence of directional light (sun light) (Fig. 7e in Appendix and Fig. 4)
3) Absence of the water body (Fig. 7f in Appendix and Fig. 4)
4) Absence of lens distortion (lens correction) (Fig. 7b in Appendix and Fig. 4)
5) Absence of colour (monochromatic images) (Fig. 7c in Appendix)
6) Replacing the water body with a plane surface (Fig. 7g in Appendix)

The first configuration depicts all the effects of the ShipSG image, making it as close to a true representation as was possible, given the constraints of this work. In configurations 2 to 5, individual effects were omitted to assess the importance

[2]Link:https://www.blender.org, accessed on 22.07.2024

[3]Link:https://www.turbosquid.com, accessed on 12.07.2024

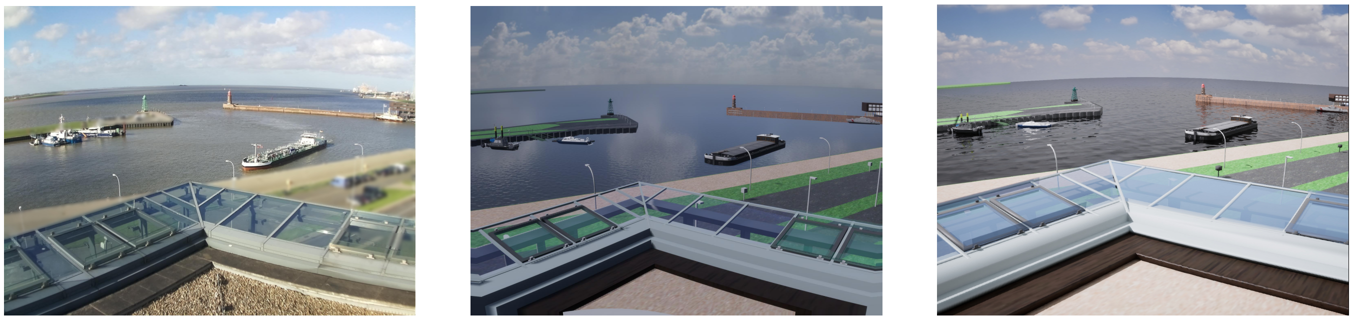


(a) Image from ShipSG dataset

(b) Key Structures in ShipSG image: Foreground - Green, Midground - Blue, Background - Yellow

(c) View of the Doppelschleuse in Blender

Fig. 2: Identifying the key structures in the ShipSG dataset



(a) ShipSG Sample Image

(b) Sample Image - Unity Engine

(c) Sample Image - Unreal Engine

Fig. 3: Example of ShipSG image with the rendered synthetic images

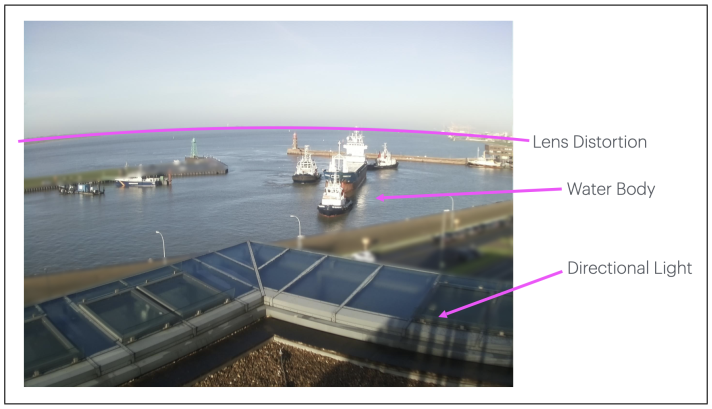


Fig. 4: Photorealism Effects in the ShipSG dataset

of each photorealistic effect, using the first configuration as the baseline. Simulating a water body is a challenging task because of the complexity involved in accurately mirroring the properties of water. The last configuration aims to study the impact of water properties and the importance of using a simulated water body. Substituting the water body with a plane surface eliminates the properties of water, such as reflection, water waves, and physics, but provides a representation of the water surface.

Each of the six scenarios were generated independently for each game engine. This approach enables us to qualitatively assess the impact of these effects on ship segmentation.

After generating 52 synthetic images for each photorealistic effect in every game engine, annotation masks are needed to evaluate the performance of the instance segmentation method. Unity engine offers an annotation toolkit called the Perception Package [16]. Unreal engine does not have its annotation toolkit. Thus, the synthetic images from the Unity engine were annotated twice, once manually, and the second time with the Perception Package. For Unreal engine, only manual annotation is utilised.

The rendered synthetic images from both the game engines replace their equivalent real images in the validation set of the ShipSG. Thus, there are three combinations with the generation and annotation of synthetic datasets:

a) images from Unity engine manually annotated (Fig. 6c)
b) images from Unity engine auto-annotated (Fig. 6a)
c) images from Unreal engine annotated manually

The six photorealistic effects described above are applied to the three combinations of synthetic dataset generation-annotation. Altogether, 18 synthetic datasets were generated to evaluate the effects that contribute to making the synthetic images photorealistic.

III. Results and Discussion

The performance of the synthetic datasets generated in Section II-C is evaluated in Section III-A. The top-performing synthetic dataset is then selected for data augmentation in Section III-B.

A. Validation Results

In reference [17], YOLOv8 [18] was utilised for ship segmentation and geo-referencing using the ShipSG dataset. YOLOv8 is a state-of-the-art real-time computer vision model built upon previous YOLO (You Only Look Once) [19] versions. YOLOv8 architecture provides support for object detection and instance segmentation amongst other features. YOLOv8 offers customisation of the architecture along with five model sizes. The models range from the fastest and lightest to the deepest and most precise: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. In this work, YOLOv8x has been chosen for its accuracy and high performance on the ShipSG dataset.

Mean Average Precision (mAP) is a widely used metric in computer vision to evaluate the precision of a model's object detection and instance segmentation capabilities. It is computed by averaging the precision across all classes, where precision is defined as the ratio of true positive detections over the sum of true positive and false positive detections. True positives are determined by whether the Intersection over Union (IoU) between the predicted and ground truth bounding boxes or instance masks exceeds a certain threshold. The IoU for instance segmentation, is calculated as:

$$IoU = \frac{area(M_{pred} \cap M_{gt})}{area(M_{pred} \cup M_{gt})} \tag{1}$$

where M_{pred} and M_{gt} represent the predicted and ground truth masks, respectively.

The Average Precision for each class i is then computed by summing the precisions at different IoU thresholds:

$$\text{AP}_i = \sum_{t \in T} p(t)\Delta t \tag{2}$$

In this formula, T is the set of IoU thresholds (normally from 0.5 to 0.95), $p(t)$ is the precision at threshold t, and Δt is the difference between consecutive IoU thresholds (normally 0.05).

Finally, the mAP is the mean of the AP values for all $N_{classes}$ classes:

$$mAP = \frac{1}{N_{classes}} \sum_{i=1}^{N_{classes}} \text{AP}_i \tag{3}$$

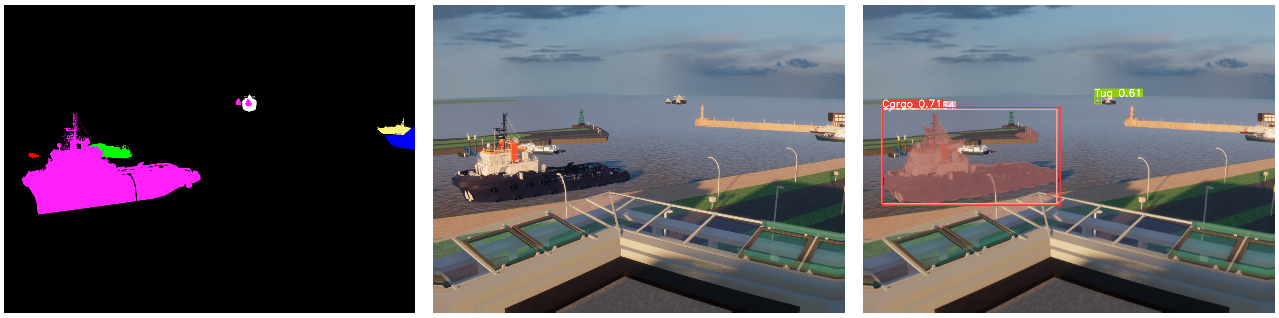
Instance segmentation using YOLOv8x performed on the ShipSG for the segmentation and classification of ships yields a mean Average Precision (mAP) of 76.5% [17]. This performance value serves as the benchmark for the evaluation of synthetic dataset performance. The weights trained on the real ShipSG training set are now used to validate the generated sets which contain the above described synthetic images. A summary of the validation result is seen in Table I. As it can be seen from Table I, none of the synthetic datasets surpass the mAP 76.5% of the real dataset, because the YOLOv8x was trained and validated on real images. This experiment aims not to match the baseline mAP, but rather to identify the synthetic image configuration that achieves the mAP closest to that of the real dataset. In this context, Unity Engine with manual annotation with all effects present and without lens distortion performs the best. Both achieve a mAP of 72.3%. The performance of Unreal Engine is consistently high, but it does not surpass Unity Engine's best. Unity Engine's performance with auto annotation experiences a decline of nearly 10% compared to the ones annotated manually, highlighting the significant role annotations play. Unity Engine's auto annotation toolkit provides pixel-precise annotation but also annotates the vessel hull that is submerged in water (Fig. 6a). The ground truth annotations (Fig. 5a) in the reference model took into account only the portion of the vessels which were above water and so the two annotation styles (Fig. 5a and 5c) are not mutually compatible. Therefore, the automatic annotation scheme is not suitable for this work.

Table I highlights the significance of incorporating photorealistic effects in synthetic images. The results consistently demonstrate that more realistic images achieve better mAP, thereby confirming that photorealism is crucial for the performance of the associated model. Removal of the directional light (sun light), changing the chromaticity by making the images monochrome and the removal of the lens distortion does not improve the performance of the synthetic datasets. The removal of the water body reduces the mAP across all three dataset types. The improvement in the performance (in comparison to the absence of a water body) when a plane surface is introduced as a replacement for the water surface, showcases the importance of a water body in synthetic maritime dataset. Thus, Unity Engine dataset, with all photorealistic effects and without lens distortion, emerges as the clear standout in this experiment.

B. Data Augmentation Experiment

For data augmentation experiment, Unity Engine dataset, with all photorealistic effects and without lens distortion are chosen. The goal of this experiment is to determine whether the inclusion of synthetic data when training the models leads to a higher mAP.

The synthetic images from the best performing synthetic dataset are added to the training set of the ShipSG dataset. Since the synthetic images replicate 52 images from the real



(a) Ground Truth - Auto Annotation (b) Sample Image (c) YOLOv8x Predicted Mask

Fig. 5: Ground Truth and Predicted Mask - Unity Engine

Configuration	**mAP (%)**		
	Unreal (manual ann.)	**Unity (auto ann.)**	**Unity (manual ann.)**
All effects	71.7	62.7	**72.3**
No directional light	71.5	62.0	69.3
No water body	71.5	62.7	69.3
No lens distortion	71.3	63.7	**72.3**
Monochromatic	71.0	62.3	69.6
Plane surface for water body	72.0	62.2	69.6

TABLE I: Validation results of YOLOv8x trained on the synthetic datasets

validation set, the corresponding real images from the validation set are removed. Consequently, the training set comprises of 2856 images (2804 original + 52 synthetic), while the validation set contains 649 images (instead of the original 701). Additionally, the annotations of the synthetic images have been added to the training set. The experiment is repeated two times, once with each of the best candidate obtained from the previous experiment. YOLOv8x is trained with the two new synthetic datasets under the same initial conditions presented in [17]. This allows for a direct comparison of the training results.

YOLOv8x training	**mAP (%)**
Original ShipSG [17]	76.5
Unity Engine (manual ann.& all effects)	**76.9**
Unity Engine (manual ann. & no lens distortion)	76.7

TABLE II: Results of the Data Augmentation Experiment after training YOLOV8x with and without synthetic images.

Following the training of the YOLOv8x on the new datasets, the configuration with all effects present exhibits an increase of 0.4% (Table II) compared to the performance of the ShipSG without the synthetic images. The dataset without the lens distortion presents an increase of 0.2% mAP. The modest increase indicates that including additional synthetic data during model training can lead to higher mAP scores, demonstrating that synthetic data has the potential to significantly enhance results for real-world maritime computer vision.

IV. Conclusion

In this paper we compare the effectiveness of game engines in generating synthetic datasets for the improvement of maritime computer vision applications, particularly focusing on the comparison between Unity Engine and Unreal Engine. Alongside this, an exploration was conducted to study the impact of various photorealistic effects on making synthetic images look realistic. A true to scale model of the Doppelschleuse was handcrafted in Blender. The crafted model was set up in the respective game engines, along with the corresponding vessel models to recreate 52 images from the validation set of the ShipSG dataset. Unity Engine's manually annotated validation set with all photorealistic effects and without lens distortion performs the best in this exploration. The mAP achieved by these two synthetic datasets stands the closest to that of the ShipSG. Unreal Engine's performance is high consistently but it isn't able to surpass Unity Engine's best. The significant difference between Unity Engine with automatic annotation and manual annotation shows the importance of consistent data labelling practices in the domain of computer vision and machine learning. The data augmentation experiment showcases the potential of synthetic datasets for maritime computer vision applications. In conclusion, the Unity Engine outperforms the Unreal Engine in generating synthetic maritime datasets for ship recognition tasks, thereby contributing to the enhancement of maritime awareness.

V. Future Work

The primary use case for synthetic datasets was to eliminate the complications that arise during acquisition. Complex scenarios (like a stormy weather) should therefore be additionally simulated. Investigating these aspects will help create more complex and diverse training datasets crucial for improving the robustness of the model. A notable insight from this work is the impact of annotations on synthetic datasets. The auto annotation toolkit reduces the time and effort significantly but needs to be modified to allow custom annotation schemes which can be tailored to match the wishes or specific needs of the user. Another pivotal area for future research is the importance of water physics in reconstructing real scenarios.

The work presented a positive outcome with a few synthetic images. To validate this result, future studies should experiment by augmenting a larger set of synthetic images into datasets.

References

[1] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 1993–2016, 2017.

[2] B. Kiefer, D. Ott, and A. Zell, "Leveraging synthetic data in object detection on unmanned aerial vehicles," 2021.

[3] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[4] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, pp. 43–77, 02 1994.

[5] S. Baker and R. Szeliski, "A database and evaluation methodology for optical flow," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE Computer Society, 2007. [Online]. Available: https://www.microsoft.com/en-us/research/publication/a-database-and-evaluation-methodology-for-optical-flow/

[6] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[8] J. Bubenicek, "Using game engine to generate synthetic datasets for machine learning," in *Proceedings of the Central European Seminar on Computer Graphics*, 2020.

[9] L. Laux, S. Schirmer, S. Schopferer, and J. Dauer, "Build your own training data - synthetic data for object detection in aerial images," in *Software Engineering 2022 Workshops*. Bonn: Gesellschaft für Informatik e.V., 2022, pp. 182–190.

[10] Unity Technologies, "Unity engine." [Online]. Available: https://www.unity.com

[11] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[12] B. Carrillo-Perez, S. Barnes, and M. Stephan, "Ship segmentation and georeferencing from static oblique view images," *Sensors*, vol. 22, no. 7, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/7/2713

[13] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 1993–2016, 2017.

[14] Z. Shao, W. Wu, Z. Wang, W. Du, and C. Li, "Seaships: A large-scale precisely annotated dataset for ship detection," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2593–2604, 2018.

[15] Google Maps, "Doppelschleuse, Bremerhaven," 2024, last Accessed: 14-07-2024. [Online]. Available: https://maps.app.goo.gl/fdp6B7ySnwHGoqKs8

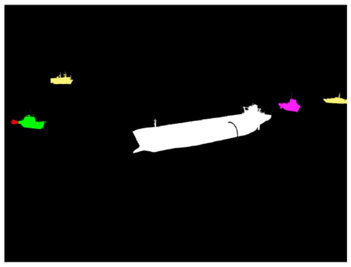
[16] Unity Technologies, "Perception package," 2020. [Online]. Available: https://github.com/Unity-Technologies/com.unity.perception

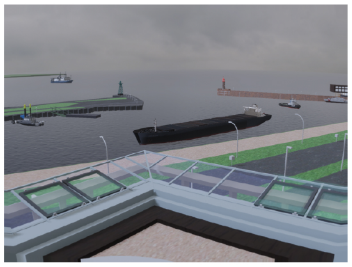
[17] B. Carrillo-Perez, A. B. Rodriguez, S. Barnes, and M. Stephan, "Improving YOLOv8 with scattering transform and attention for maritime awareness," in *2023 International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2023, pp. 1–6.

[18] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[19] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

Appendix
Sample images of the synthetic datasets


(a) Auto Annotation


(b) Sample Image

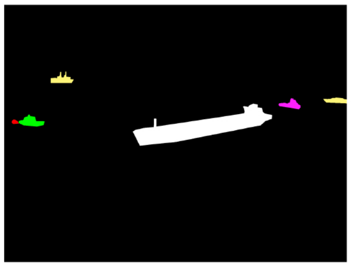
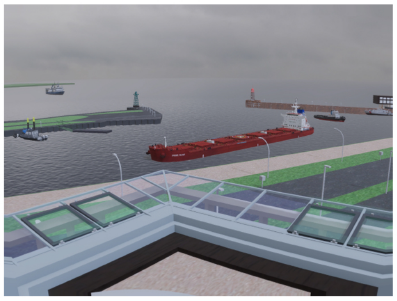
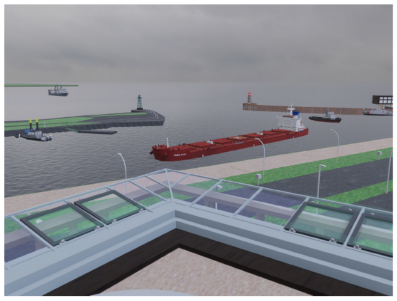

(c) Manual Annotation

Fig. 6: Visual comparison of manual and automatic annotation of ship masks on the Synthetic image. The colors of the masks represent different ship categories. - Unity Engine

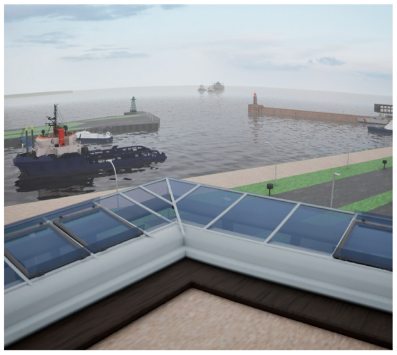

(a) All Effects Present

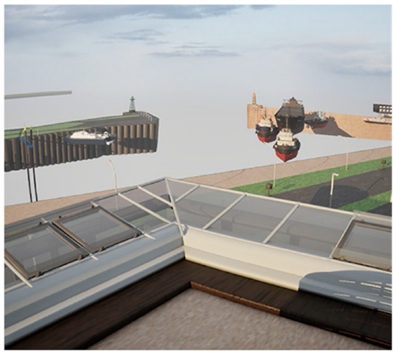

(b) No Lens Distortion

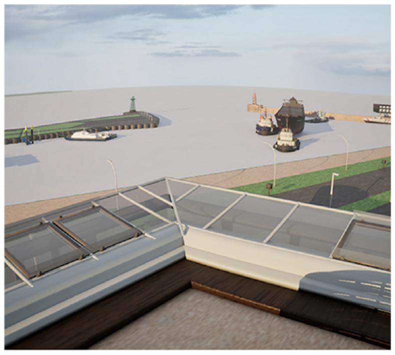
(c) Monochrome Image

Unity Engine

(d) With Directional Light


(e) No Directional Light


(f) No Water Body


(g) Plane Surface for Water Body

Unreal Engine

Fig. 7: Generated Images with various photorealistic effects - Unity & Unreal Engine