A Novel Framework to Evaluate and Train Object Detection Models for Real-Time Victims Search and Rescue at Sea with Autonomous Unmanned Aerial Systems Using High-Fidelity Dynamic Marine Simulation Environment

Rajeev Poudel Luciano Lima
University of South-Eastern Norway
rajeev_poudel@yahoo.com, limaluciano@ieee.org

Fabio Andrade
University of South-Eastern Norway
NORCE Norwegian Research Centre
fabio@ieee.org

Abstract

This work presents a novel framework providing the ability to control an Unmanned Aerial System (UAS) while detecting objects in real-time with visible detections, containing class names, bounding boxes, and confidence scores, in a changeable high-fidelity sea simulation environment, where the major attributes like the number of human victims and debris floating, ocean waves and shades, weather conditions such as rain, snow, and fog, sun brightness and intensity, camera exposure and brightness can easily be manipulated. Developed using Unreal Engine, Microsoft AirSim, and Robot Operating System (ROS), the framework was firstly used to find the best possible configuration of the UAS flight altitude, and camera brightness with high average prediction confidence of human victim detection, and then only autonomous real-time test missions were carried out to calculate the accuracies of two pretrained You Only Look Once Version 7 (YOLOv7) models: YOLOv7 retrained on SeaDronesSee Dataset (YOLOv7-SDS) and YOLOv7 originally trained on Microsoft COCO Dataset (YOLOv7-COCO), which resulted in high values of 97.8% and 93.79%, respectively. Furthermore, it is proposed that the framework developed in this study can be reverse engineered for autonomous real-time training with automatic ground-truth labeling of the images from the gaming engine that already has all the details of all objects placed in the environment for rendering them onto the screen. This is required to be done to avoid the cumbersome and time-consuming manual labeling of large amount of synthetic data that can be extracted using this framework which could be a groundbreaking achievement in the field of maritime computer vision.

1. Introduction

Unforeseeable in nature, disasters involving ships at sea not only inflict costly economic and environmental damage, but also jeopardize the invaluable life of crew and passengers onboard. According to [20], there were a total of 892 shipping losses worldwide between 2012 to 2021 with 54 total mishaps alone in 2021. Even though the total number of global vessel hazards declined by around 57% over the decade, it is still a substantial amount with each case necessitating prompt and costly deployment of Search and Rescue (SAR) teams to rapidly curb down the resulting harm. And, naturally, the primary focus of all rescue missions is to first scour the inhospitable post-disaster region for victims and safeguard their lives. All this substantiates the research interest to effectively and efficiently utilize the existing cutting-edge scientific innovations to alleviate the threat on human life emanating from unpredictable maritime accidents.

However, the abundance of all the applicable contemporary technologies introduces perplexity in deciding the perfect combination between them for optimum performance. In general, almost all major research conundrums are resolved with the thorough comprehension of the problem domain and taking inspirations from the phenomenon already occurring in nature. On breakdown of present real-life search and rescue operations, intuitively most of the associated expense including time and money is attributed to the transportation of human first responders in boats, helicopters, and aircrafts [16]. In addition, the involvement of humans, pursuant to [3], brings upon various errors due to estimation biases of different physical quantities such as under-estimation of horizontal distance, over-estimation of height when looking down and under-estimation when looking up. These drawbacks can be overcome using Unmanned Aerial Systems (UAS) that have small-size, lower operational cost, flexible aerial maneuverability, wireless communication, and mathematical computation ability. UAS equipped with simple RGB and/or thermal cameras and either onboard or cloud-based processing capability which facilitates the use of deep convolutional neural networks (CNN) based object detection models, as discussed by the authors in [25], [27], [18], [14], [11] and [17], can best mimic the action of rescue personnel flying in helicopters or aircrafts for finding the victims in hazardous territories, making the rescue process more efficient. Furthermore, among different modern deep learning based object detection models [26], the state-of-the-art YOLOv7 that transcends all other recognized object detectors in speed and accuracy [23] is here considered the most suitable one because in critical real-time SAR missions both response time and accuracy are equally important for saving human life. Therefore, the starting scientific dilemma is now narrowed down to the paramount research question that forms the main basis for this work which is: How to find the best possible configurations of the UAS and state-of-the-art object detection models for working together in real-time with optimal accuracy of victims detection at an erratic post-disaster ship scenario?

With this question in mind, simulation seems to be the only plausible path forward initially because of the risk, price, time, and effort involved to set up the physical test environment at sea with real persons and UAS with cameras, not to mention the absurd complications in the re-enactment of the alternating scenario in the aftermath of an actual ship accident. Moreover, the general prerequisites of the simulation platform to be used can also be deduced from the research question as: (1) It should be able to produce detailed reproduction of a disaster-struck ship surroundings with high quality of graphics; (2) It should allow the replica of UAS with various sensors to be spawned and controlled in the fabricated environment; (3) It should have an interface to a mechanism capable to control as well as read and process sensor data from a real UAS, and execute object detection models, enabling transferability to real-world applications; and (4) It should have the ability to pass a continuous image stream from the replicated UAS that can be fed as input to object detection models for real-time processing.

Unreal Engine 4 [7] with the integration of AirSim [19], and Robot Operating System (ROS) [21], on the basis of [5], [2], [15], [12] and [24], has the potential to fulfill all the requirements of the simulation platform for this work as mentioned above. But when the requirements are actually materialized with the combination of Unreal Engine, AirSim, ROS, and Object Detection Models, a novel framework originates that answers the research question.

Therefore, this work follows the steps according to the requirements to firstly develop the framework. Then, using this framework, the object detection models are evaluated to find the finest configurations for achieving high accuracy of victim detection in real-time.

Hence, the main contributions of this paper are summarized as:

- The creation of a high-fidelity changeable sea simulation environment, where the deep-rooted challenges in the maritime computer vision such as the different light conditions, altitudes, sea colors, buoyancy, objects movement, camera exposures, brightness, weather, size of the objects, among many others, can be easily controlled. This also allows to inexhaustibly generate synthetic data for training new models.

- The development of a framework with the constructed simulation environment to evaluate the performance of the cutting-edge object detection models with the input images from the UAS in real-time autonomous SAR missions, which can directly be transferred to real-world UAS applications.

- The proposal to reverse engineer the created framework for autonomous real-time training of object detection models with the automatic ground-truth labeling of the desired objects in the images from the UAS which could be a breakthrough in maritime computer vision.

2. Development of the Novel Framework

This section describes the overall steps carried out based on the requirements of the simulation platform mentioned in the previous section.

2.1. Virtual Environment

In this section, all the steps to build the simulated environment will be presented.

The simulation environment is composed of a oil tanker, objects and people in water, and a small boat where the drones are deployed from.

In Figure 1, the environment is presented, highlighting the oil tanker. Another angle of the environment, highlighting the objects and people can be seen in Figure 2.

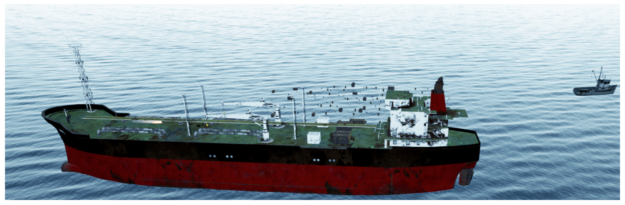


Figure 1: Simulated environment from oil tanker side.

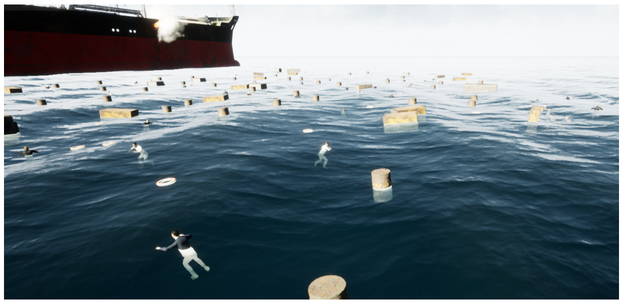


Figure 2: Simulated environment from objects side.

2.1.1 Environment Project

The EnvironmentProject [6] is an open source environment simulation project for Unreal Engine 4. It is the continuation of the OceanProject, and has many features, such as ocean simulation, sky simulation, buoyancy, time and fish plugins. In this work, the simulation environment was built on top of an existing example world that is made available by the EnvironmentProject.

Two important configurations that are only present when building sea environments are the color of the ocean and the waves. It is possible to choose a darker or brighter ocean or more blue or green, for example. Regarding the waves, it is possible to choose the height, direction, among others, to make a more stormy or calm sea. In the EnvironmentProject world, these configurations are in the Blueprint ”BP_Ocean”. Additionally, the various environmental aspects like sunlight intensity, brightness, atmospheric light, fog, and others were present in the blueprint ”BP_Sky”.

In addition, it is possible to configure weather parameters such as wind, rain, among others, which are also present in any world of Unreal Engine 4, but have their own plugin in the EnvironmentProject.

2.1.2 Post-Disaster Oil Tanker

The first element that was added in the environment was a post disaster ship.

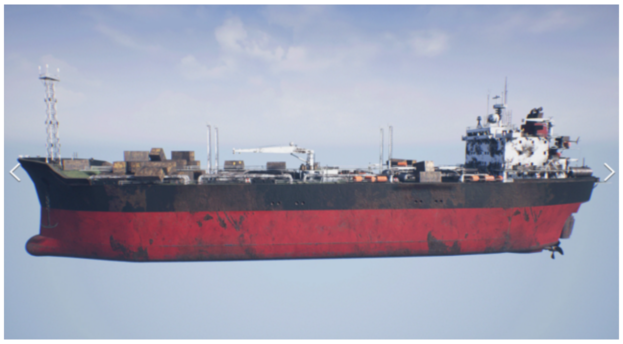


Figure 3: Post Disaster Oil Tanker.

Unreal Engine 4 Marketplace has much content available for download, both free and paid. The content that was chosen for this work is called ”Post-Apocalyptic Oil Tanker” and was made available for purchase in 2017 by the content creator ”mikkotahtinen”. An illustration of the ship can be seen in Figure 3. It is important to note that the content that is downloaded is composed by many separate blueprints (Figure 4). The creator of the world needed to build the oil tanker with the desired content. One advantage was that in the content there were many other interesting objects such as containers, that were added in the environment developed by this work.

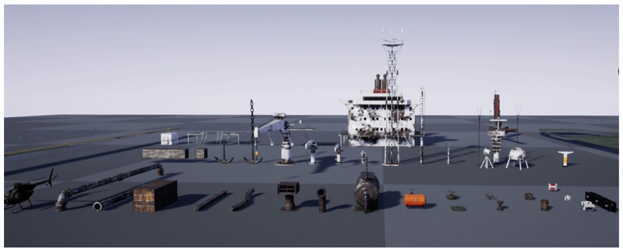


Figure 4: Blueprints available in the ”Post-Apocalyptic Oil Tanker” product.

2.1.3 People

As the goal of this proposed framework was to provide a realistic environment, it was required to populate it with people. This work focused on including people treading water to simulate victims in a sea disaster. However, it is also possible to include people walking in the ship or swimming. Many characters and animations can be downloaded for free at Mixamo [1] by Adobe.

In this work, around six different characters were used, all of them with the animation of treading water.

After downloading the animation, the physics aspects must be properly configured. The two configurations that allow the person to properly tread water and be affected by the water movement are to enable collision and choose the ”SK_Mannequin_PhysicsAsset” as the ”Physics Asset Override”. This was implemented with the proper understanding of similarity in the bone structure and hierarchy of the ”SK_Mannequin” Asset which is the default third person character of Unreal Engine, and the Mixamo character. This also allows the manual control of the Mixamo characters using the physics control capability of the ”SK_Mannequin”.

2.1.4 Buoyancy Configuration

One of the main aspects of this work is to have objects which are affected by the stream and waves of the environment. Therefore, the buoyancy must be correctly configured, otherwise, the objects would just be with a static position, frozen in the 3D space, without following the water movement.

To configure the buoyancy, first the ”BuoyantForce” component was added to the Blueprint, then, the buoyancy points were decided with the assistance of the arrow tool. Therefore, it is possible to know the exact position to add the buoyancy in the ”Test Points” configuration element. For the swimmer, three buoyancy points were added. This varies for different objects.

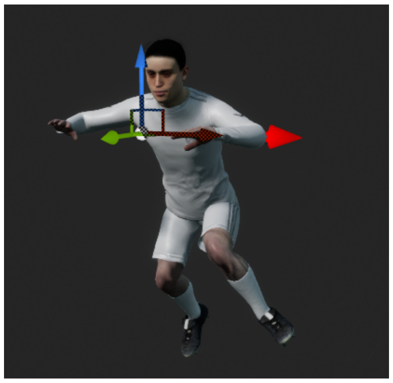
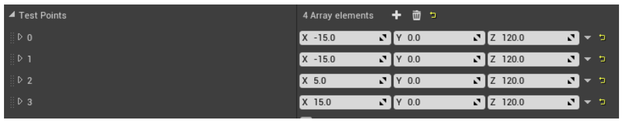


Figure 5: Buoyancy points configuration.

Finally, Figure 6 presents the three test points which were included for the swimmer blueprint.



Test Points	4 Array elements		
0	X -15.0	Y 0.0	Z 120.0
1	X -15.0	Y 0.0	Z 120.0
2	X 5.0	Y 0.0	Z 120.0
3	X 15.0	Y 0.0	Z 120.0

Figure 6: Buoyancy points placement.

It is important to note that the same procedure must be performed for all objects placed on the sea, such as the oil tanker, containers,oil barrels, buoys, among others. Nevertheless, the buoyancy points should be added to only one blueprint of any object, and then the same object can be easily replicated with the same settings.

2.1.5 Other aspects

In addition, buoys and other objects, such as containers and oil barrels with buoyancy added following the same procedure as people, were placed as seen in Figure 7.

2.2. Initial Setup of the UAS

Firstly, the AirSim plugin was integrated into the custom Unreal environment following the procedures as explained in the AirSim documentation [13]. All settings, except for camera, were kept as default. A single multirotor UAS named ”Drone_1” was spawned in the environment

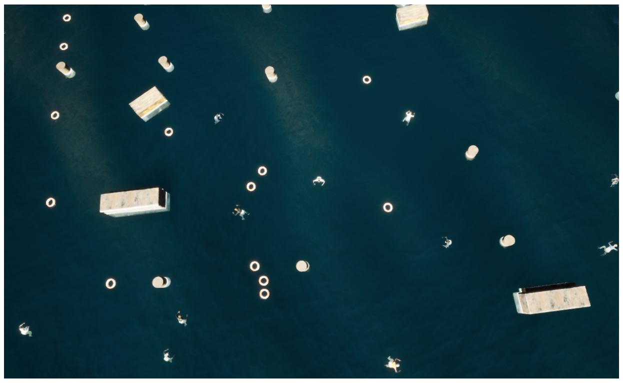


Figure 7: Top view with objects.

with ”PlayerStart” placed on top of a rescue fishing boat as shown in Figure 8.

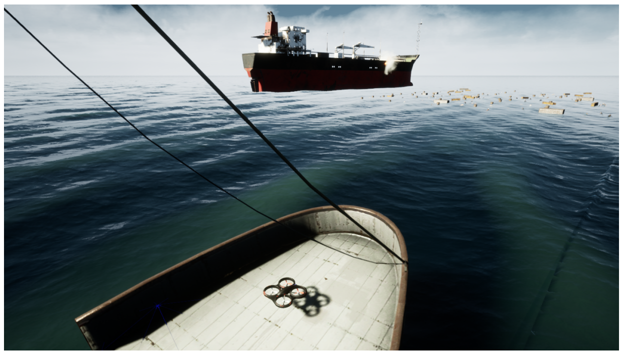


Figure 8: Initial UAS Setup in the Virtual Environment.

The camera settings were modified facilitating the UAS to have a single camera of resolution 640x640, which is the YOLOv7 model standard image resolution, field of view (FOV) of 90 degrees, and gimbal enabled with perfect stabilization of 1 and pitch of -90 degrees making the camera face vertically downward. In addition, the sensors like IMU, Magnetometer, GPS, and Barometer were also enabled automatically if the settings were left unchanged for the Multirotor sim mode as mentioned in the AirSim documentation [13].

Furthermore, complying with the directives specified, Airsim ROS wrapper was setup for Noetic version of ROS inside Windows Sub-system for Linux (WSL) 2 with Ubuntu 20.04 as Linux distribution on a Windows 10 computer having NVIDIA GeForce RTX 2080 Ti Graphical Processing Unit (GPU). It primarily contained two nodes among which the mostly used first node named ”airsim_node” was a wrapper over AirSim's multirotor C++ client library that was comprised of various publishers, subscribers, services, and parameters.

2.3. Manual Control of the UAS

Next, using the ”Twist” ROS message type, the velocity command subscriber topic from the wrapper allowed the movement of the UAS in all directions with the input of both linear and angular velocities in x, y, and z coordinates. For utilizing this feature to manually move the UAS in a desired way in the simulation environment, a ROS package named ”AS_RoS_Teleop” was used that linked the different keyboard keys with separate control commands to publish velocity twist messages in the chosen topic.

2.4. Implementation of YOLOv7 in ROS

Subsequently, the effort of implementing YOLOv7 in ROS was eased with the ready-made ROS package titled ”yolov7_ros” which was a ROS wrapper built over the original framework by the official developers of YOLOv7 [23]. After that, the weights of the chosen pretained YOLOv7 models were downloaded, and the class names for the respective models in the required txt file format were saved in separate folders. Then, the path to the model weights, class names, and the image topic were specified accordingly in the launch file to initiate the node for the real-time detection and visualization of the detections along with the bounding boxes, class names, and confidence scores using the desired YOLOv7 model one at a time.

3. Evaluation of Object Detection Models

This section explains the different procedures adopted to evaluate the performance of the object detection models for real-time detection of human victims in autonomous UAS missions.

3.1. Selection of Pretained YOLOv7 Models on Different Datasets for Evaluation

As this study was in its early phase, it was decided to utilize the ready-to-use YOLOv7 models that were already trained on datasets containing people because the focus of this study was to detect human victims with high accuracy in the post-disaster scenarios.

The first obvious choice was the originally trained YOLOv7 model on Microsoft COCO (Common Objects in Context) [10] which was a large-scale dataset developed for object detection, classification and segmentation with 91 labeled objects constituting also people designated as ”person” class. Due to the core nature of any Deep CNN based models including YOLOv7 to learn patterns in the training image using shifting convolution operations, it was important to assess the type of human images in this dataset. So on further scrutiny, it was found that the majority of the images were taken in canonical perspective [8] with different viewing angles.

Secondly, in search of datasets specially concentrating on the marine environment and aerial images, SeaDronesSee [22] was found, which was also a large-scale dataset from different aerial perspectives developed with focus on SAR operations on the sea using UAS. This was completely relevant for this work. In addition, the SeaDronesSee team had also trained YOLOv7 in their own dataset, and made the model freely available in project GitHub [9]. The output labels in this model were swimmer (people floating with stretched hands and legs), boat, jetski, buoy, and life saving appliance (life jacket/lifebelt).

3.2. Experimentation with Various Configurations

The main beauty of the developed framework was that it enabled numerous experiments with minimal efforts which otherwise would have been either impossible or extremely difficult in real-life.

However, to make the study more focused in accordance with all other experimental studies, the variables to be considered in this work were also reduced from the plethora of the manipulable variables. Thus, keeping constant the environmental factors such as dark blue ocean shade, low wave amplitude and velocity, normal level of atmospheric and other lights, only the UAS position, especially height, and camera brightness was manipulated. The camera brightness was altered by changing the post-process settings present inside the camera component of the main parent blueprint of AirSim Camera named ”BP_PIPCAMERA”. Also, to further reduce the variables involved in this study, the camera brightness was changed as very low, low, normal, high, and very high. When the environment is executed in AirSim Game Mode, the images rendered on the screen are from the external camera which is also a child of the parent AirSim Camera. Hence, when the brightness of the camera was changed,it affected the image displayed on the viewport as seen in 10.

Therefore, the starting experiment was carried out by freely traversing the UAS in the environment with different camera brightness and YOLOv7 models. On doing so, some interesting phenomenon of human victim detection were observed for both the models.

With the YOLOv7-SDS model selected, all the objects were detected as ”boat” class in low or normal camera brightness for all heights of the UAS. But when the brightness was high, the model started to detect floating people with hands and legs moving as ”swimmers” whereas other objects still as ”boat”. Meanwhile, with the YOLOv7-COCO model chosen, the human buoyant victims were correctly classified as ”person” class mostly in low heights with low or normal brightness.

For concretizing these observations, a separate test area with just the imported six characters was created as shown in Figure 9.

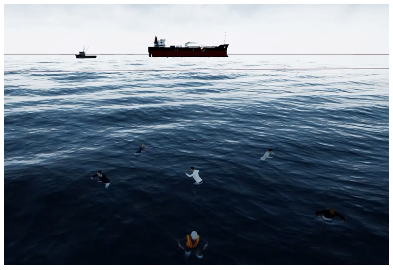


Figure 9: Isolated test region with just humans.

After that, the UAS was manually flown to the centre of the testing region, and slowly only the altitude of the UAS was elevated from low to high and vice versa with different camera brightness each time for both YOLOv7-SDS and YOLOv7-COCO models. Concurrently, the detections with bounding boxes and confidence scores, the average prediction confidences and the altitude were closely monitored as shown in Figure 10.

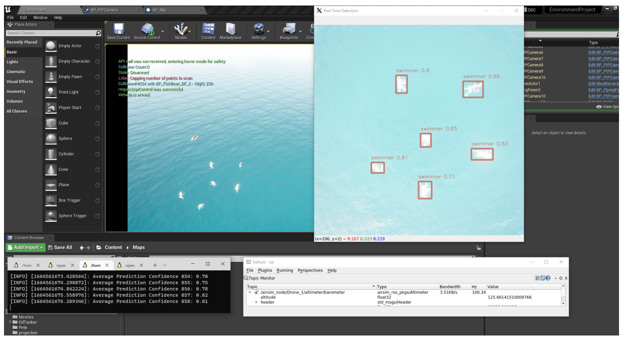

```
[INFO] [1664561673.428564]: Average Prediction Confidence 854: 0.70
[INFO] [1664561674.298872]: Average Prediction Confidence 855: 0.75
[INFO] [1664561674.842224]: Average Prediction Confidence 856: 0.78
[INFO] [1664561675.558976]: Average Prediction Confidence 857: 0.82
[INFO] [1664561676.289346]: Average Prediction Confidence 858: 0.81
```

Figure 10: Illustration of the experimental procedures followed in the testing region.

Finally, after exhaustive trials it was found that the YOLOv7-SDS model had the highest average prediction confidence of detecting human victims as ”swimmers” class at 8 meters from the sea level with a very high camera brightness, whereas the the YOLOv7-COCO model had the highest average prediction confidence of detecting human victims as ”person” class at 2 meters from the sea level with low camera brightness.

3.3. Path Planning for the Autonomous Mission

As the primary objective of this study was to evaluate the performance of the models for victim detection by skipping the arduous process of deploying the UAS in actual post-disaster scenarios with a simulated one, there was a need to replicate the mission that would have been employed in real-life, which could be used to gather the test images after detection by the models for empirical accuracy calculation.

Moreover, the predetermination of the specific height and camera settings of the UAS also laid the foundation for the autonomous surveillance mission. Using the distributed node processing capability of the ROS framework, the responsibilities of taking the UAS to the appropriate location in the environment, and then covering the desired locality fully were assigned to separate nodes. The point-to-point transfer of the UAS was implemented by modifying the second node present in the AirSim ROS wrapper named ”Simple PID Controller Node” from service node into an action server node waiting for the position goal asynchronously where the controller parameters proportional gain (K_p) and derivative gain (K_d) were set after heuristic tuning to 0.5, and 2 respectively.

For full coverage of the desired post-disaster region by the UAS, the boustrophedon path [4], as shown in Figure 11, was deemed to be the most straightforward and effective option for this work, where the width in each step was selected to be:

$$width = 2 \times Z_{UAS} \times \tan\left(\frac{FOV}{2}\right), \tag{1}$$

where Z_{UAS} is the altitude of the UAS and FOV is the field of view of the camera.

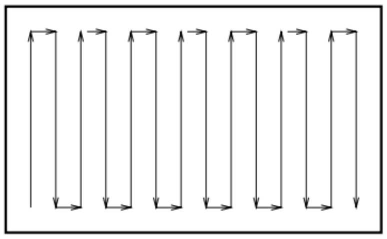


Figure 11: Boustrophedon Path.

Here, the boustrophedon path was implemented in a node where the odometry sensor topic was subscribed for current UAS position and the required velocities were published to the velocity command topic in world frame.

Furthermore, an additional path planner node was created for the autonomous systematic execution of both the nodes.

3.4. Final Mission Execution

Lastly, the autonomous reconnaissance operations were carried out, where the drone independently takes off, goes to the specified starting point of the desired area, covers the area for predefined mission time, and returns back to land in the initial position. All these actions were executed by the collaboration between the different nodes discussed in the previous sections as shown in Figure 12.

Also, using the ”image_view” package in ROS, the real-time images with detections published during the mission by the ”yolov7_ros” node in the visualization topic were observed and some chosen images at strategic locations con-

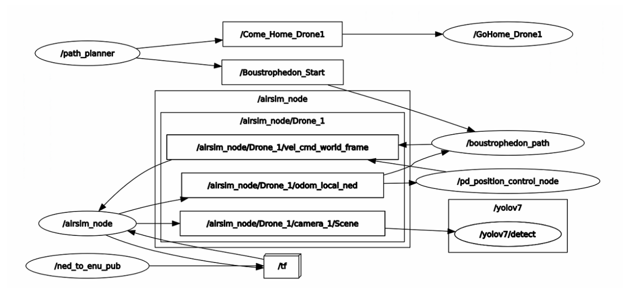


Figure 12: Collaboration between the nodes during the mission obtained using ”rqt_graph”.

taining people were saved by simple right-click of mouse-button for both the models which ensured proper representative sampling for statistical calculation. This was mainly done to limit the number of images gathered for numerical testing due to high frame rate without losing substantial features from the data.

4. Results and Discussion

This section presents the results from the statistical metrics calculation of the selected output test images from the real-time detection models along with the relevant discussions.

Table 1 illustrates the flight altitude and camera brightness of the UAS for each model in addition to the number of test images gathered for evaluation. The number of images sampled was lower when using the YOLOv7-SDS because the flight elevation was higher, and as each time the altitude is doubled, the area covered is quadrupled, so less images were needed for the representative analysis of the same area.

Model	Altitude	Brightness	Images
YOLOv7-COCO	2 m	Low	130
YOLOv7-SDS	8 m	Very High	26

Table 1: Overview of the datasets collected and assessed.

The confusion matrix of the detections in the YOLOv7-COCO model can be seen in Table 2. There were 129 objects detected. Of these 129 objects, 110 were correctly detected as ”person”, 2 were incorrectly classified as another object and 6 people were not detected at all. Therefore, the number of false negatives is equal to 8. That gives an accuracy of 93.79%. Among the correctly detected people, the average confidence level was of 84%, the minimum was 27% and the maximum 96%.

The confusion matrix of the detections in the YOLO-SDS model can be seen in Table 3. There were 142 objects detected. Of these 142 objects, 36 were correctly detected as swimmers, there were 1 incorrectly classified as other object and 2 swimmers were not detected at all. Therefore, the number of false negatives is equal to 3. That gives an accuracy of 97.8%. Among the correctly detected swimmers, the average confidence level was of 69%, the minimum was 38% and the maximum 82%.

		Predicted	
		Person	Not Person
Actual	Person	110	8
	Not Person	0	11

Table 2: Confusion Matrix of the detections in the collected images for the Yolov7 model trained on the COCO2017 dataset (YOLOv7-COCO).

		Predicted	
		Swimmer	Not Swimmer
Actual	Swimmer	36	3
	Not Swimmer	0	103

Table 3: Confusion Matrix of the detections in the collected images for the Yolov7 model trained on the SeaDronesSee dataset (YOLOv7-SDS).

Now, the first point to be discussed is the very fact that the models trained on real images were able to detect the synthetic objects with high accuracies provides a strong proof-of-concept for the interchangeability of real and virtual SAR missions, justifying the importance of this work.

Secondly, although the accuracies achieved by both models were high, there was a huge difference in the nature of the input images fed into the models. This inspired further contemplation on the working of the deep CNN itself. As, in deep learning the patterns in pixel level from the input images are encoded into the model, so the behavior of the model is dependent on the normalized pixel intensities in the three RGB color channels which is actually the numerical input into the model. With this comprehension, the results from the YOLOv7-model, as seen in Figure 14, made complete sense because the patterns in training images of MS COCO Dataset [10] matched with the pixel-level patterns in the input images due to the adjusted flight altitude of the UAS. But, the result from the YOLOv7-SDS, as observed in Figure 13, was a surprising discovery which was only possible due to the rigorous experiments with various configurations of altitude and camera settings allowed by this framework. Comparing this result to the images with people in the SeaDronesSee Dataset [22], it could only be hypothesized that the matching of the pixel-level pattern of a human floating in water with stretched hands and legs along with high uniform intensities of the bright pixels triggered the neural network to output ”swimmer” class. This intriguing phenomena needs more research and can be particularly interesting for training with high altitude or satellite images.

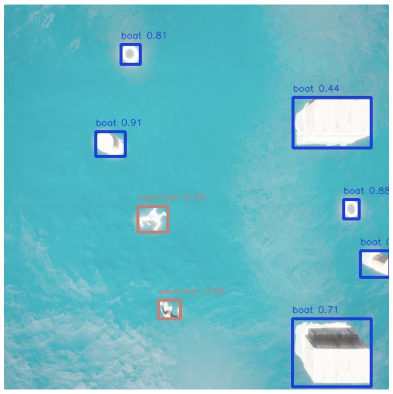
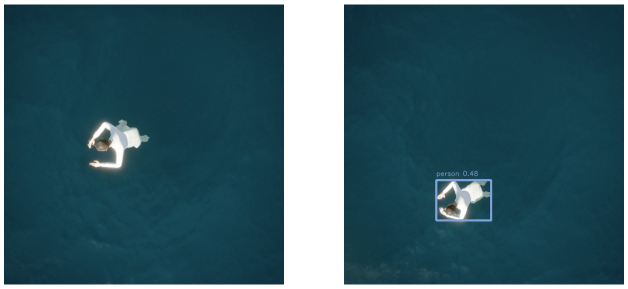


Figure 13: Detection with YOLOv7-SDS in mission.

Lastly, another captivating observation highlighting the significance of real-time detection in SAR is that with a high frame rate of input images, the footprint underneath the UAS is always overlapping with the UAS moving forward only small distance in each iteration resulting in multiple chances of detection. So, even if a victim is not detected in one image, there is still a high probability that it might have been detected in the earlier or will be detected in the subsequent image which is clearly exemplified by Figure 14. This also results in the actual false omission rate (FOR) being lower than what was observed with the test images.



(a) Frame 40 (b) Frame 41

Figure 14: Detections with YOLOv7-COCO in mission.

5. Conclusion

In this work, a framework was developed using Unreal Engine, Microsoft AirSim, and ROS that enabled the control of the UAS in a desired way based on velocity commands while detecting objects in an alternating post-disaster ship simulation environment. After that, two pretained YOLOv7 models were selected: YOLOv7-SDS and YOLOv7-COCO. Then, using the created framework, extensive manual testing was implemented by changing the UAS altitude along with the camera brightness to discover the best possible combination with high average prediction confidence of detecting human victims naturally floating in water with moving hands and legs. This allowed to plan and implement autonomous UAS missions using the same framework yielding high accuracies of real-time victim detection calculated as 97.8% for YOLOv7-SDS when the UAS was deployed in an altitude of 8m with high camera brightness, and 93.79% for YOLOv7-COCO when the UAS was employed at a height of 2m from the sea-level with lower camera brightness.

Furthermore, the developed framework has immense potentiality for further work. Due to the limitation of time and difficulty of accommodating all the things in a single paper, only few experiments were carried out in this study. But extensive experimentations will be performed in near future with various object detection models in different configurations accompanied by the implementation of the findings physically with a real UAS. Additionally, other researchers are also encouraged to utilize the detailed steps of reproducing the framework fabricated in this work to carry out experiments according to their respective needs.

Moreover, a small modification in the research question proposed in this work to "How to make *any* configuration of the UAS and object detection models to work together in real-time with optimal accuracy of detecting *any* desired object in a dynamic marine environment?" suddenly demystifies the true potential of this framework. By reverse engineering the images in the UAS camera to have the precise automatic ground truth labeling from the gaming engine, the same framework has the capability of training new object detection models in real-time in different imaginable configurations for any thinkable objects overcoming all the traditional challenges in maritime computer vision.

Acknowledgment: This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No. 101020676 for the project named as VALKYRIES (Harmonization and Pre-Standardization of Equipment, Training and Tactical Coordinated procedures for First Aid Vehicles deployment on European multi-victim Disasters).

References

[1] Adobe. Mixamo by adobe. `http://www.mixamo.com`. Accessed: 2022-10-14.

[2] Fabio Augusto De Alcantara Andrade, Agnar Holten Sivertsen, Carlos Alberto Moraes Correia, Nabil Belbachir, Lucas Costa De Sousa, Victor Müller Pereira Rufino, Eduardo Pimenta Petrópolis, Erick Rodrigues e Silva, and Victor Hugo Rinaldi Fortes Henriques. Virtual reality simulation of autonomous solar plants inspections with unmanned aerial systems. In *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*, pages 1–8. IEEE, 2021.

[3] Yee-Yin Choong, Gavriel Salvendy, et al. Voices of first responders–applying human factors and ergonomics knowledge to improve the usability of public safety communications technology, 2021.

[4] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209. Springer, 1998.

[5] Tuan Do Trong, Quan Tran Hai, Nam Tran Duc, and Han Trong Thanh. A novelty approach to emulate field data captured by unmanned aerial vehicles for training deep learning algorithms used for search-and-rescue activities at sea. In *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*, pages 288–293. IEEE, 2021.

[6] DotCam, TK-Master, Zoc, and Sascha Elble. Environmentproject. `https://github.com/UE4-OceanProject/Environment-Project`, 2022.

[7] Epic Games. Unreal engine. `https://www.unrealengine.com`. Accessed: 2022-10-24.

[8] James Higgins. *Canonical views of objects and scenes*. University of Illinois at Urbana-Champaign, 2011.

[9] Benjamin Kiefer. Seadronessee. `https://github.com/Ben93kie/SeaDronesSee`, 2022.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[11] Josip Lorincz, Adnan Tahirović, and Biljana Risteska Stojkoska. A novel real-time unmanned aerial vehicles-based disaster management framework. In *2021 29th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2021.

[12] Chenxiang Ma, You Zhou, and Zhiqiang Li. A new simulation environment based on airsim, ros, and px4 for quadcopter aircrafts. In *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pages 486–490, 2020.

[13] Microsoft. Airsim. `https://microsoft.github.io/AirSim/`. Accessed: 2022-10-24.

[14] Balmukund Mishra, Deepak Garg, Pratik Narang, and Vipul Mishra. Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, 156:1–10, 2020.

[15] Anne Redulla and Surya PN Singh. Simulating differential games with improved fidelity to better inform cooperative & adversarial two vehicle uav flight. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 130–136. IEEE, 2018.

[16] Regjeringen. The norwegian search and rescue service. `https://www.regjeringen.no/globalassets/upload/kilde/jd/bro/2003/0005/ddd/pdfv/183865-infohefte_engelsk.pdfl`. Accessed: 2022-10-24.

[17] Christopher Dahlin Rodin, Luciano Netto de Lima, Fabio Augusto de Alcantara Andrade, Diego Barreto Haddad, Tor Arne Johansen, and Rune Storvold. Object classification in thermal images using convolutional neural networks for search and rescue missions with unmanned aerial systems. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[18] Sasa Sambolek and Marina Ivasic-Kos. Automatic person detection in search and rescue operations using deep cnn detectors. *IEEE Access*, 9:37905–37922, 2021.

[19] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

[20] Allianz Global Corporate & Specialty. Safety and shipping review 2022. `https://www.agcs.allianz.com/news-and-insights/reports/shipping-safety.html`. Accessed: 2022-10-24.

[21] Stanford Artificial Intelligence Laboratory et al. Robot operating system. `https://www.ros.org/`.

[22] Leon Amadeus Varga, Benjamin Kiefer, Martin Messmer, and Andreas Zell. Seadronessee: A maritime benchmark for detecting humans in open water, 2021.

[23] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[24] Shubo Wang, Jian Chen, Zichao Zhang, Guangqi Wang, Yu Tan, and Yongjun Zheng. Construction of a virtual reality platform for uav deep learning. In *2017 Chinese Automation Congress (CAC)*, pages 3912–3916. IEEE, 2017.

[25] Shubo Wang, Yu Han, Jian Chen, Zichao Zhang, Guangqi Wang, and Nannan Du. A deep-learning-based sea search and rescue algorithm by uav remote sensing. In *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, pages 1–5. IEEE, 2018.

[26] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing*, page 103514, 2022.

[27] Ruidong Zheng, Ruigang Yang, Keqiao Lu, and Shesheng Zhang. A search and rescue system for maritime personnel in disaster carried on unmanned aerial vehicle. In *2019 18th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 43–47. IEEE, 2019.