SeaDronesSee: A Maritime Benchmark for Detecting Humans in Open Water – Supplementary Material

Leon Amadeus Varga*, Benjamin Kiefer*, Martin Messmer*, Andreas Zell
Cognitive Systems Group
University of Tuebingen
Tuebingen, Germany
Email: leon.varga@uni-tuebingen.de, benjamin.kiefer@uni-tuebingen.de,
martin.messmer@uni-tuebingen.de, andreas.zell@uni-tuebingen.de

1. Further Statistics

In this section, we provide statistics that may provide further insights into SeaDronesSee.

1.1. Object Detection

Figure 1 shows the distribution of instance diameters in the object detection task. Most of the object are slightly below 50 pixels in size.

To assess the variability in sizes of instances, we compare to other data sets by following the convention in [14] and [13] which measure the size of an object by its horizontal bounding box length. Furthermore, we also form three groups of instances according to their size: 0-50 px, 50-300 px, and >300 px. Table 1 shows the distribution over these different groups. As also noted in [13], PASCAL VOC, NWPU VHR-10 and Munich Vehicle are dominated by medium-sized and small-sized objects. In contrast, DOTA and SeaDronesSee offer a more balanced distribution between small sized and middle sized objects. This favors models benchmarked on our data set that perform well on all object sizes. For instance, a model should detect the largest object, having 900 pixels of width, and the smallest object, having 5 pixels of width, which is 180 times smaller.

We note that these values are not normalized to the image sizes. Furthermore, the thresholds are taken from [14]. Different thresholds may yield different numbers. We chose these for comparability.

1.2. Object Tracking

To assess the distribution of frames for the object tracking task, we plot the angle-altitude distribution in a 2D-histogram in Figure 2. For most of the altitude-angle-domains, there are more than 1,000 frames. For higher altitudes, the angles become more acute since the objects would become too small otherwise (triangular shape). On the contrary, for lower altitudes footage at all angles are uniformly captured.

The class distribution and instance diameter distribution is very similar to the one for object detection which is why we omit it here. How images are distributed on different cameras is shown in Figure 3. Note that the RedEdge-MX and UMC-R10C are not shown since they only produced still images.

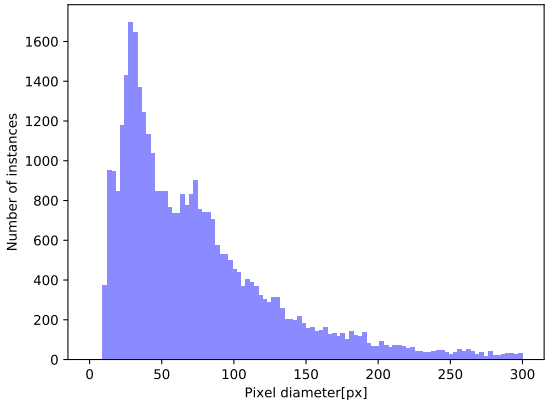


Figure 1. Distribution of instance diameters in the object detection task. Bounding boxes larger than 300 pixels are not shown for visualization purposes.

Dataset	10-50px	50-300px	>300px
PASCAL VOC [11]	0.14	0.61	0.25
NWPU VHR-10 [10]	0.15	0.83	0.02
Munich Vehicle [5]	0.93	0.07	0
DOTA [13]	0.57	0.41	0.02
SeaDronesSee	0.42	0.54	0.04

Table 1. Percentages of instances in respective groups which are divided according to size.

*These authors contributed equally to this work. The order of names is determined by coin flipping.

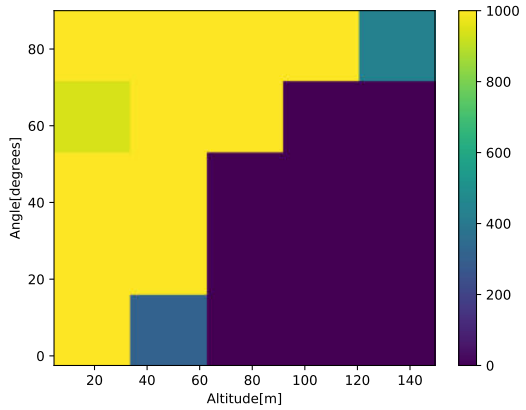


Figure 2. Number of images for specific altitude and angle domains in the tracking data set.

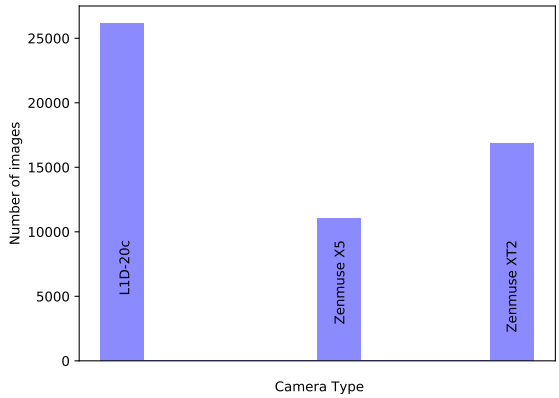


Figure 3. Number of images for specific altitude and angle domains in the tracking data set.

2. Data Set Collection - Further Information

The data acquisition mission was undertaken considering safety measurements regarding the safety during swimming and the Covid-19 regulations. In particular, the safety during swimming aspect was taken seriously as swimmers were swimming in open and deep water. All swimmers (and floaters) were overseen at all times. Additionally, a lifeguard monitored the scene at all times.

Furthermore, safety regulations regarding flying UAVs had to be taken seriously. Official permissions were needed to fly. Furthermore, as multiple UAVs were flying at the same time, a strict flight plan had to be followed. Especially, the vertical division into different flight zones helped safely fly multiple UAVs simultaneously.

The flight plans for the Quantum Systems Trinity F90+ were done in the flight planning software Qbase. With a flight time of around 90 minutes, several altitudes and x-y-coordinates could be reached autonomously. The quadcopters were initially steered using the flight plans made in Litchi. However, as the current was strong swimmer would drift away and the initial plans were useless. It turned out that it was too hard to incorporate the current as it changed quickly. We switched to manual photographs after that.

One major point to consider were environmental regulations. Surrounding wildlife areas were considered and not crossed with boat and UAV.

The meta data that was captured during the flights was evaluated using Airdata [1].

3. Annotation Guidelines

When annotating SeaDronesSee, we defined some guidelines on how to do it:

- Every bounding box should be as small as possible while still fitting the entire instance.
- No instance may be left out in an image unless the size of it is less than three pixels on either side.
- If there are ambiguous other objects (such as boats or other objects on land, unknown objects in water), they should be annotated as ignored region or blackened out.
- Objects across images should always be annotated in the same way to achieve consistency.

In the case of SeaDronesSee, we have seven classes including ignored regions. Swimmers and Swimmers with life jacket (all humans in water) were labeled such that the bounding box includes the whole body, including arms and legs, if they are visible in the water (compare to the drawing Vitruvian Man). If they are not visible but could easily be inferred, they are not labeled. Boats are annotated such that only the hull of the boat is labeled if there is no overlap between the roof and the hull. If there is an overlap, then annotate the whole boat including roof. Humans are annotated similarly as swimmers but without limbs spreading out too far. See the end of this document for examples of annotations.

4. Uploading Formats

As mentioned in the evaluation section of the main body of this work, we provide an evaluation server to prevent researchers from overfitting on our benchmark and for fair comparisons on our leaderboards. The file formats vary for different tasks which are described in detail here. Note, however, that this is subject to change and will be announced on the webserver.

4.1. Object Detection

The results on the test set of the object detection task need to be submitted as a json-file containing the predictions on the test images. The json-file is a list of dictionaries, where each list item is dictionary of the following form

imgid	Number of corresponding test set image (integer).
bbox_x	x-coordinate of the top-left corner of the predicted bounding box (float).
bbox_y	y-coordinate of the top-left corner of the predicted bounding box (float).
bbox_w	the width (in pixels) of the predicted bounding box (float).
bbox_h	the height (in pixels) of the predicted bounding box (float).
conf.	The confidence value for the bounding box. Can be a value between 0 and 1. Please see the AP metric for details on the influence of this value on the final metric value (float).
cat.	Category id; any of the following integers: 1 (swimmer), 2 (floater), 3 (boat), 4 (swimmer†), 5 (floater†), 6 (life jacket) (integer).

Table 2. Format for the predictions to upload.

```
{"image_id": imgid, "category_id": cat,
"bbox": [bbox_x, bbox_y,
         bbox_w, bbox_h],
"score": conf}
```

The value descriptions and its data format are shown in Table 2.

In particular, background bounding boxes and ignored regions shall not be uploaded or predicted.

4.2. Single-Object Tracking

In this case, for every video clip a single text file needs to be uploaded within a zip-file. The text-files should be named after the corresponding video clip without the file extension. Each line of a respective text file contains the location and size of the target in that specific frame such that each text file has as many rows as its corresponding video has frames. With the notation from before any row is formatted as follows:

```
bbox_x, bbox_y, bbox_w, bbox_h
```

4.3. Multi-Object Tracking

Each text file stores the results for its corresponding video clip. The text file should have the same name as its corresponding video clip without the file extensions. Each line of any text file is formatted as follows

$$\#, \text{id}, \text{bbox}_{\text{left}}, \text{bbox}_{\text{top}}, \text{bbox}_{\text{width}}, \text{bbox}_{\text{height}}, \text{conf.}, \text{cat.} \quad (1)$$

where entries are as above in addition to

#	Frame index in video
id	The target id

4.4. Evaluation Webpage

We built an evaluation webpage, where researchers can upload their predictions on the respective task's test set. They will be compared against the ground truth labels by computing the respective metric in real-time on the webserver. Researchers can then opt to publish their result on a central leaderboard. The leaderboard is accessible via `https://seadronessee.cs.uni-tuebingen.de`. The number of submissions per task per day is limited such that overfitting is largely avoided.

5. Multi-Object Tracking Evaluation Details

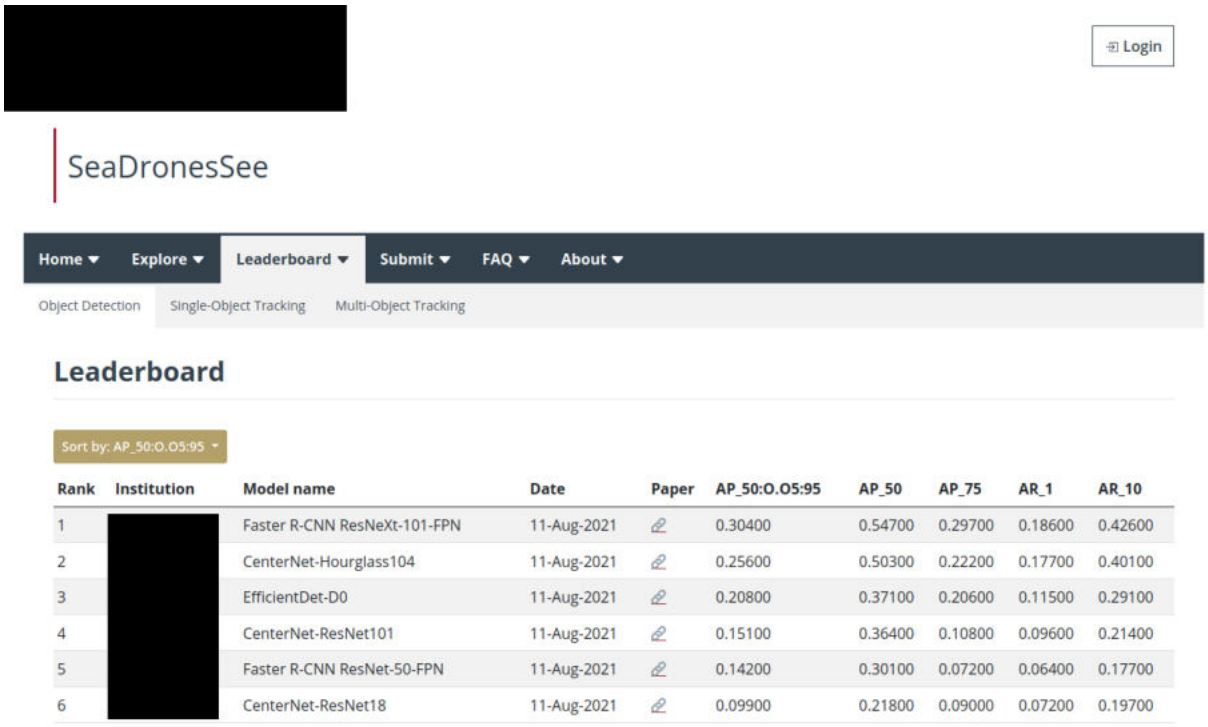
The MOTA metric combines the FP, FN and ID sw. The IDF1 score depicts the ratio of correctly classified detections over the average number of ground-truth and computed detection. The MOTP value is the average dissimilarity between all true positives and the corresponding ground-truths. ID sw. describes the total number of times that the id of an instance is switched incorrectly. Further information can be found in [9].

6. Hyperparameter and Training Configurations

All networks are implemented and trained using the PyTorch [8] framework. For all models and experiments, we used a single Nvidia RTX 2080 Ti for training and evaluation. For the EfficientDet-$D0$ we also tested its real-time applicability on a Nvidia Xavier. We make use of several popular freely and openly accessible repositories, which we will list in the following.

6.1. Object Detection

The Faster R-CNN models with ResNeXt-101-FPN and ResNet-50-FPN backbones were implemented on top of [6]. The model and training configurations and hyperparameters were adapted to provide state-of-the-art results on the VisDrone benchmark. Namely, the following changes are made to the default configurations (which can be seen in the repository): The anchor sizes and strides are decreased to $(16, 32, 64, 128, 256)$ and $(4, 8, 16, 32, 64)$ to allow for small object detection. The region of interest bounding box head pooler resolution is decreased to 7 and the pooler scales are adapted to $(0.25, 0.125, 0.0625, 0.03125)$. Training was done with SGD with learning rate 0.001 and weight decay 0.0001 for 20 epochs and another ten epochs with learning rate 0.0001.



Rank	Institution	Model name	Date	Paper	AP_50:O.O5:95	AP_50	AP_75	AR_1	AR_10
1	[redacted]	Faster R-CNN ResNeXt-101-FPN	11-Aug-2021		0.30400	0.54700	0.29700	0.18600	0.42600
2		CenterNet-Hourglass104	11-Aug-2021		0.25600	0.50300	0.22200	0.17700	0.40100
3		EfficientDet-D0	11-Aug-2021		0.20800	0.37100	0.20600	0.11500	0.29100
4		CenterNet-ResNet101	11-Aug-2021		0.15100	0.36400	0.10800	0.09600	0.21400
5		Faster R-CNN ResNet-50-FPN	11-Aug-2021		0.14200	0.30100	0.07200	0.06400	0.17700
6		CenterNet-ResNet18	11-Aug-2021		0.09900	0.21800	0.09000	0.07200	0.19700

Figure 4. Example page (leaderboard for object detection) of evaluation webpage.

We used a detectron2-based CenterNet implementation (`https://github.com/FateScript/CenterNet-better`). CenterNet was trained on an image-width of 1024 pixels. For the ResNet backbones, the initial learning rate was $1 \cdot 10^{-4}$. The learning rate was reduced by the factor of 0.1 when the validation loss didn't improve in the last 3 epochs. Further, The Hourglass-backbone was trained with the best configuration on the VisDrone-dataset of [7].

We used a detectron2-based CenterNet implementation [12]. CenterNet was trained on an image-width of 1024 pixels. For the ResNet backbones, the initial learning rate was $1 \cdot 10^{-4}$. The learning rate was reduced by the factor of 0.1 when the validation loss didn't improve in the last 3 epochs. Further, The Hourglass-backbone was trained with the best configuration on the VisDrone-dataset of [7].

We conducted our experiments involving EfficientDet with the implementation from [4]. Instead of using $\left(1, 2^{1/3}, 2^{2/3}\right)$ as the standard anchor-scales, we chose $(0.3, 0.6, 0.9)$ to account for the small objects in the data set. For the same reason we scaled every image so that its longer edge is 1280 px long (keeping the aspect ratios) instead of the usual 512 px for EfficientDet$-D0$. The IoU-thresholds, which discriminate between positive and negative training samples, were 0.15 and 0.25 instead of 0.4 and 0.5.

6.2. Single-Object Tracking

We make use of PyTracking [2] with the default configurations of Atom, DiMP50, DiMP18, PrDiMP50 and PrDiMP18.

6.3. Multi-Object Tracking

For the Tracktor++, we used the official implementation [3] with the default configuration. Only the parameters of the reid-part were reduced ($P = 6$ and $K = 4$). For our experiments involving FairMOT, we used the implementation from [15] without any alterations.

References

[1] Airdata for UAV. `https://app.airdata.com/`. Accessed: 2021-03-01. 2

[2] Pytracking - python framework for visual object tracking. `https://github.com/visionml/pytracking`. Accessed: 2020-11-01. 4

[3] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 4

[4] Signatrix GmbH. A pytorch implementation of efficientdet object detection. *https://github.com/signatrix/efficientdet*, 2020. 4

[5] Kang Liu and Gellert Mattyus. Fast multiclass vehicle detection on aerial images. *IEEE Geoscience and Remote Sensing Letters*, 12(9):1938–1942, 2015. 1

[6] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. `https://github.com/facebookresearch/maskrcnn-benchmark`, 2018. Accessed: [2020-11-01]. 3

[7] Dheeraj Reddy Pailla, Varghese Kollerathu, and Sai Saketh Chennamsetty. Object detection on aerial imagery using CenterNet. 2019. 4

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 3

[9] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer, 2016. 3

[10] Hao Su, Shunjun Wei, Min Yan, Chen Wang, Jun Shi, and Xiaoling Zhang. Object detection and instance segmentation in remote sensing imagery based on precise mask r-cnn. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 1454–1457. IEEE, 2019. 1

[11] Sara Vicente, Joao Carreira, Lourdes Agapito, and Jorge Batista. Reconstructing pascal voc. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 41–48, 2014. 1

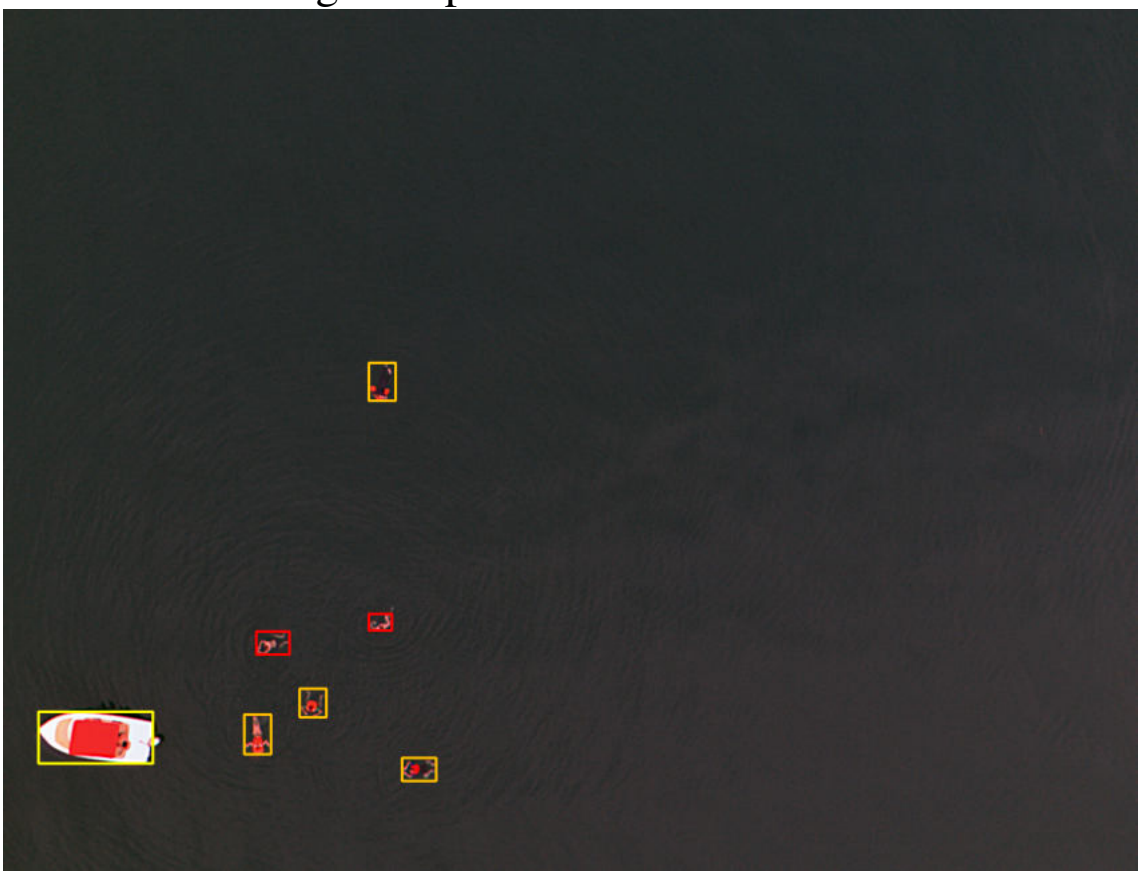
[12] Feng Wang. Centernet-better. `https://github.com/FateScript/CenterNet-better`, 2020. 4

[13] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3974–3983, 2018. 1

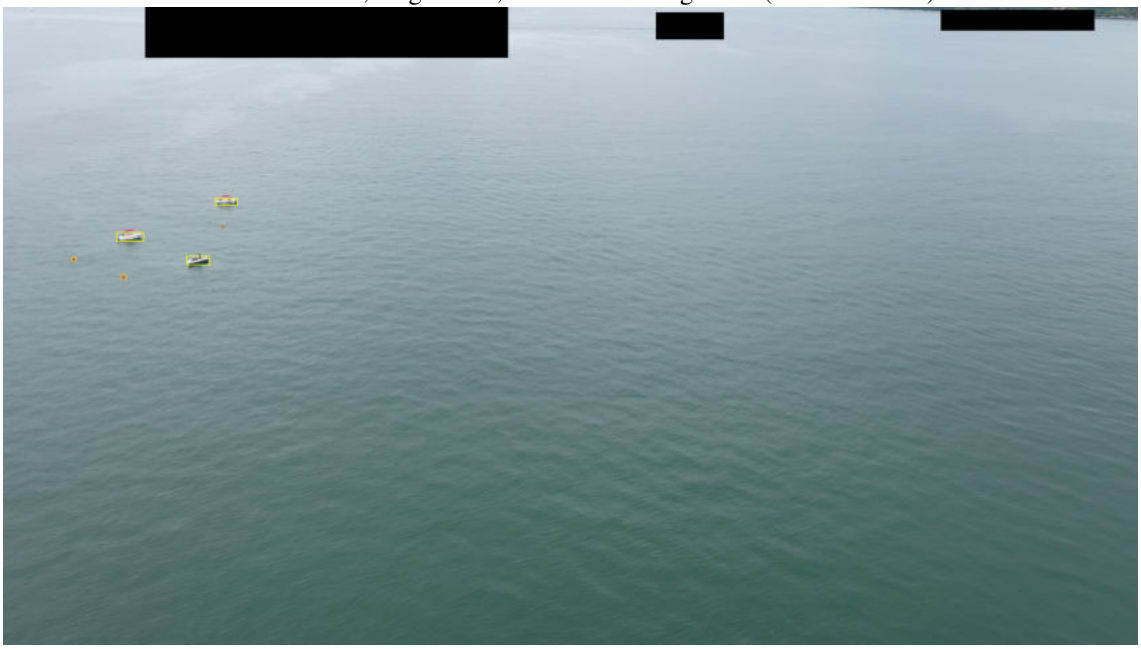
[14] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016. 1

[15] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020. 4

Image samples from SeaDronesSee

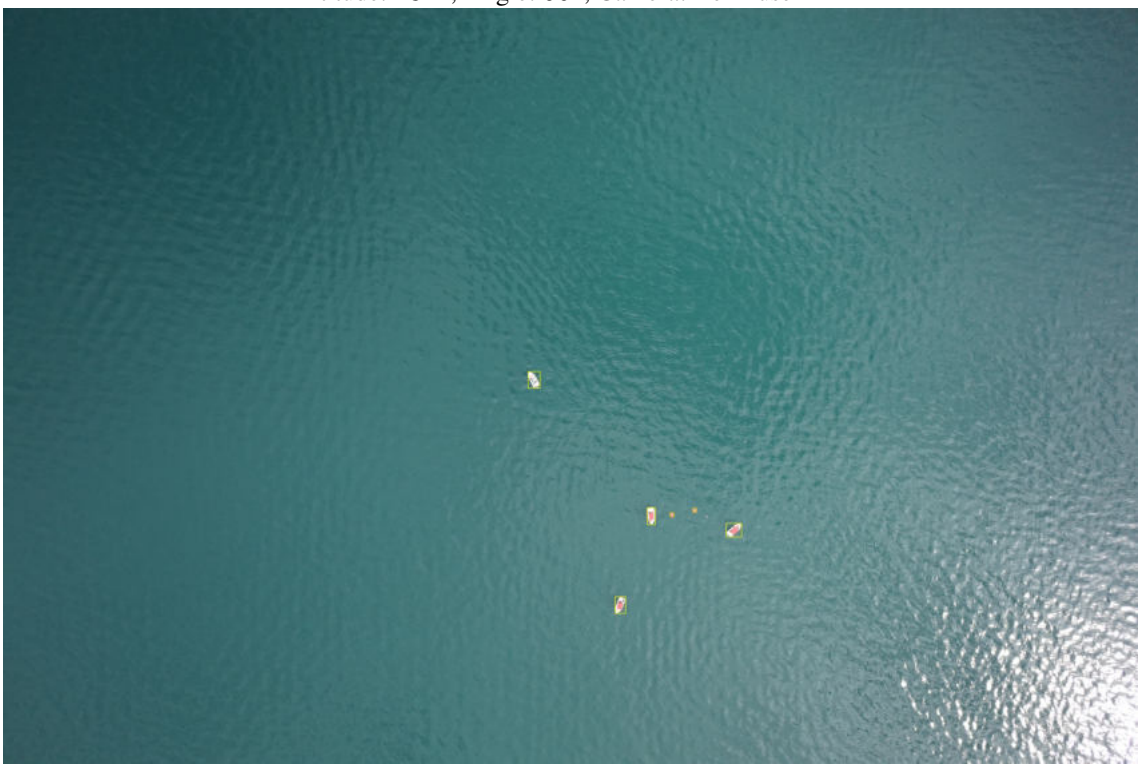


Altitude: 57 m; Angle: 90°; Camera: RedEdge-MX (RGB channels)



Altitude: 50 m; Angle: 20°; Camera: L1D-20C

Altitude: 23 m; Angle: 90°; Camera: Zenmuse XT2



Altitude: 229 m; Angle: 90°; Camera: UMC-R10C

Altitude: 41 m; Angle: 87°; Camera: Zenmuse X5

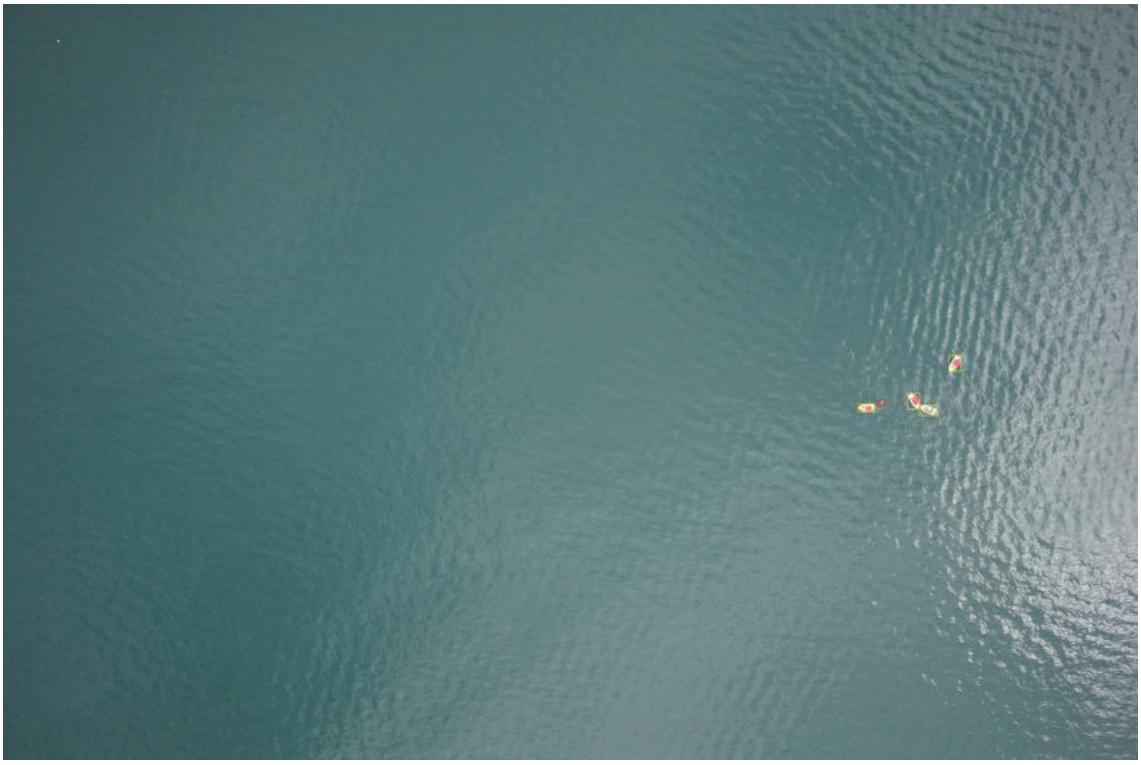
Altitude: 9 m; Angle: 45°; Camera: L1D-20C

Altitude: 20 m; Angle: 16°; Camera: L1D-20C

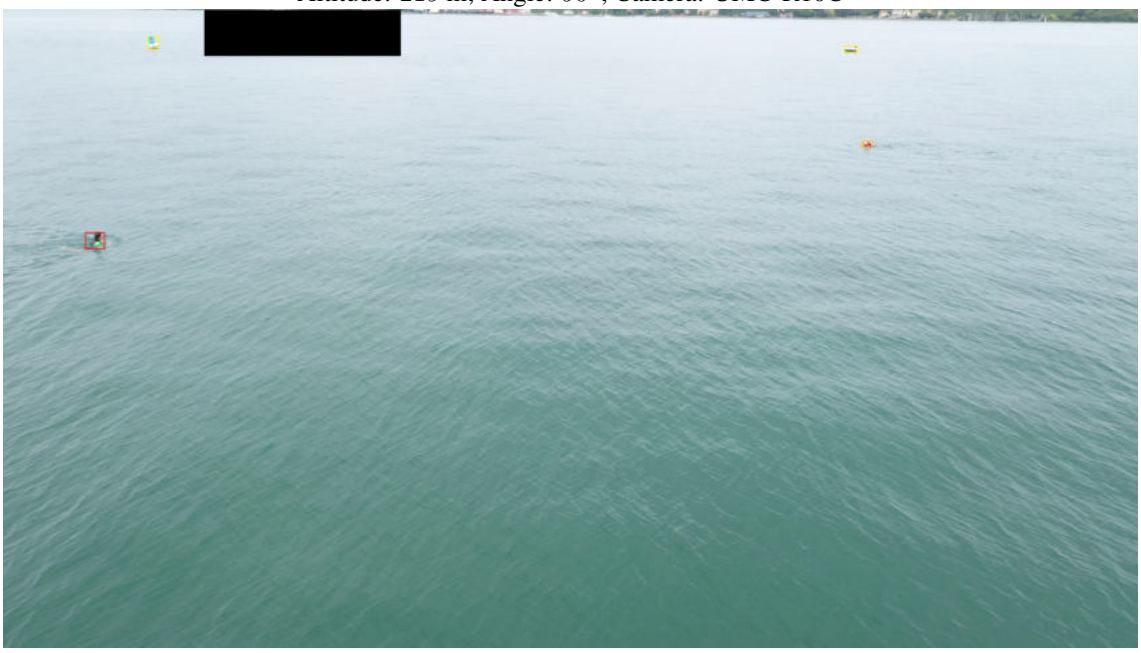
Altitude: 128 m; Angle: 90°; Camera: UMC-R10C

Altitude: 128 m; Angle: 90°; Camera: UMC-R10C

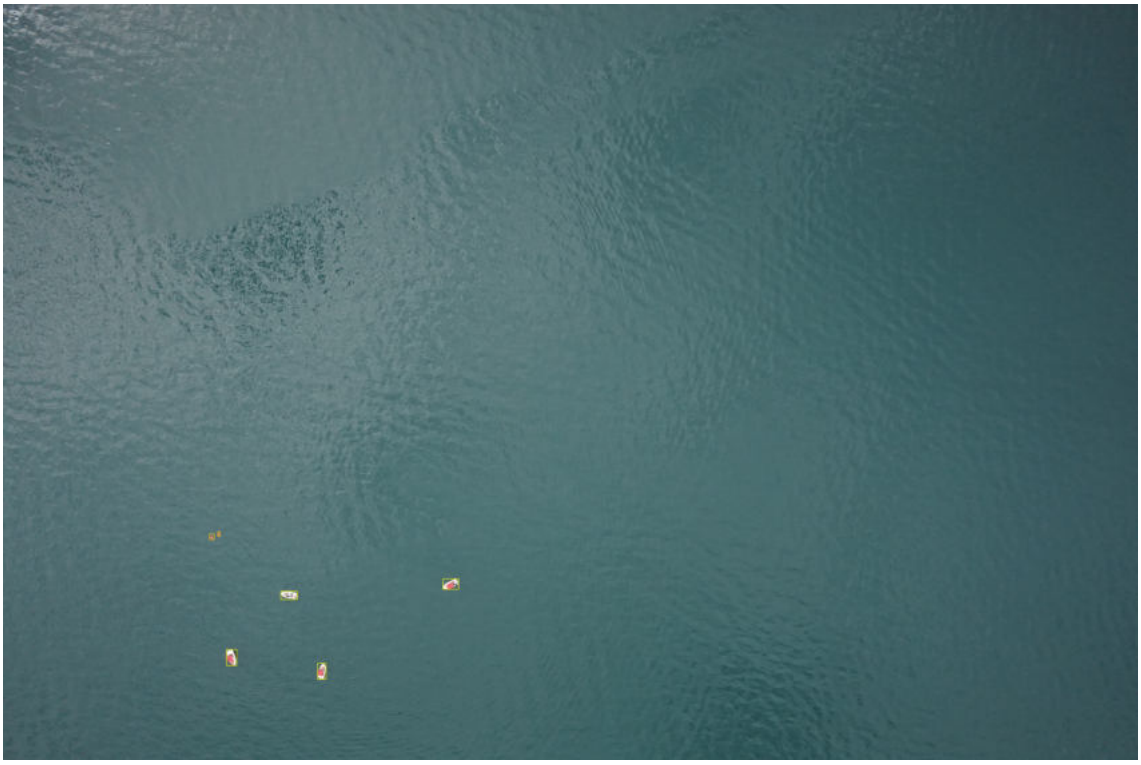
Altitude: 58 m; Angle: 90°; Camera: UMC-R10C



Altitude: 219 m; Angle: 90°; Camera: UMC-R10C



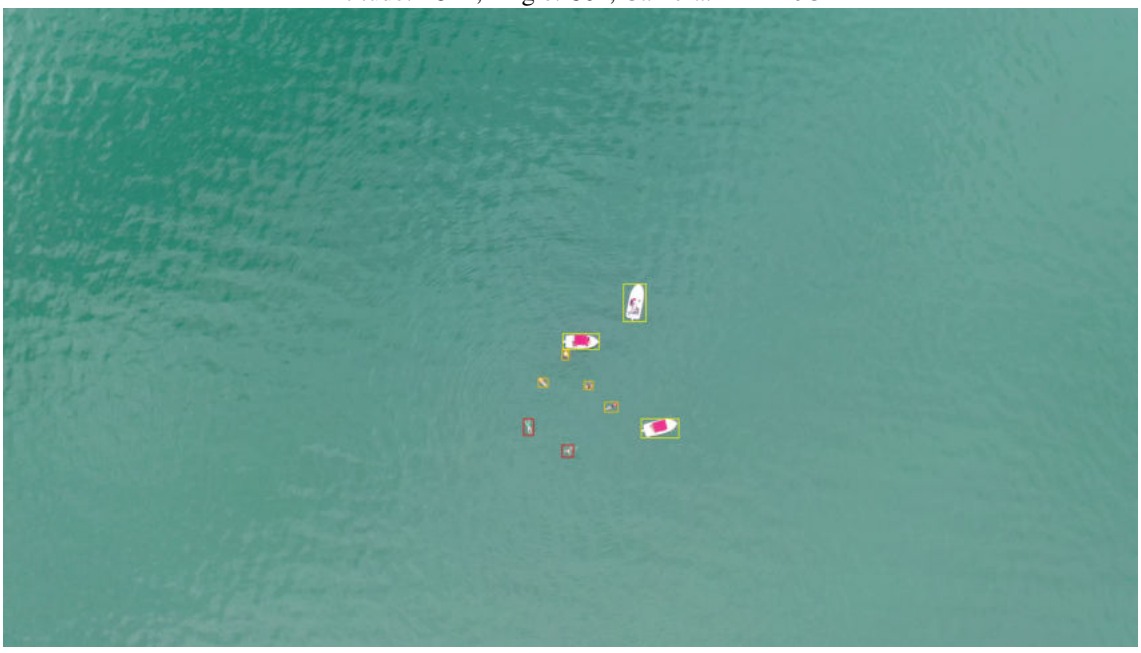
Altitude: 10 m; Angle: 24°; Camera: L1D-20C



Altitude: 237 m; Angle: 90°; Camera: UMC-R10C

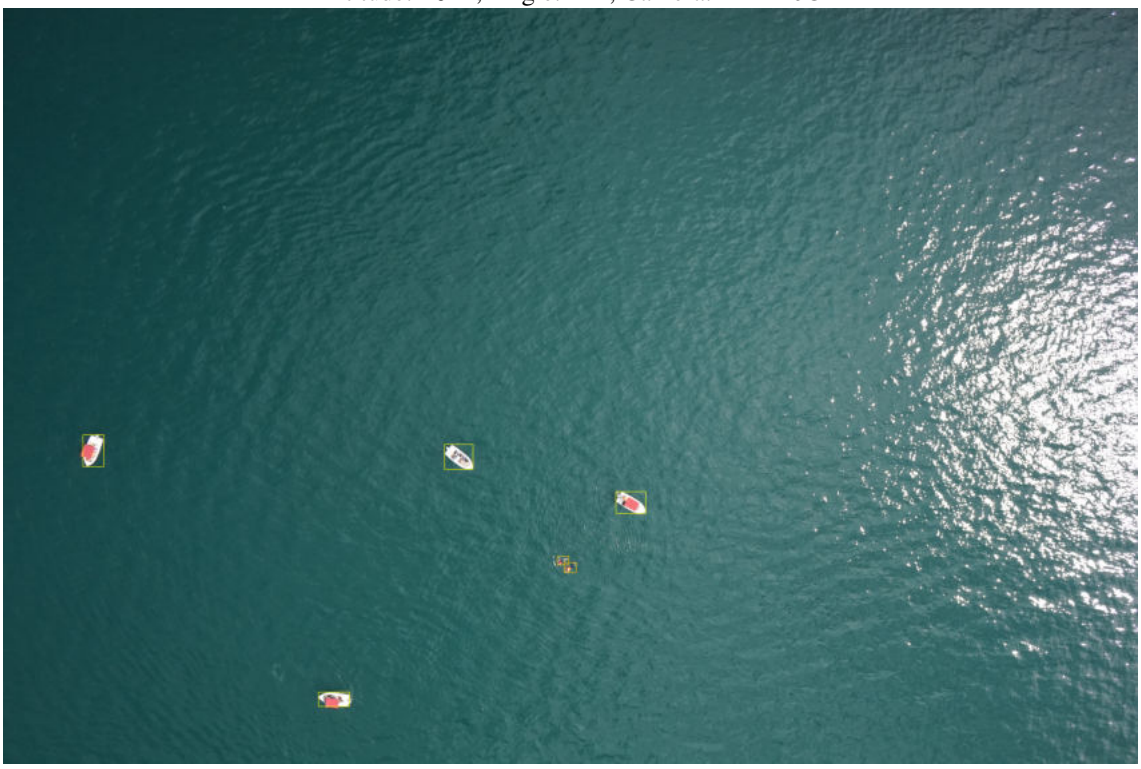
Altitude: 40 m; Angle: 34°; Camera: L1D-20C

Altitude: 48 m; Angle: 36°; Camera: L1D-20C



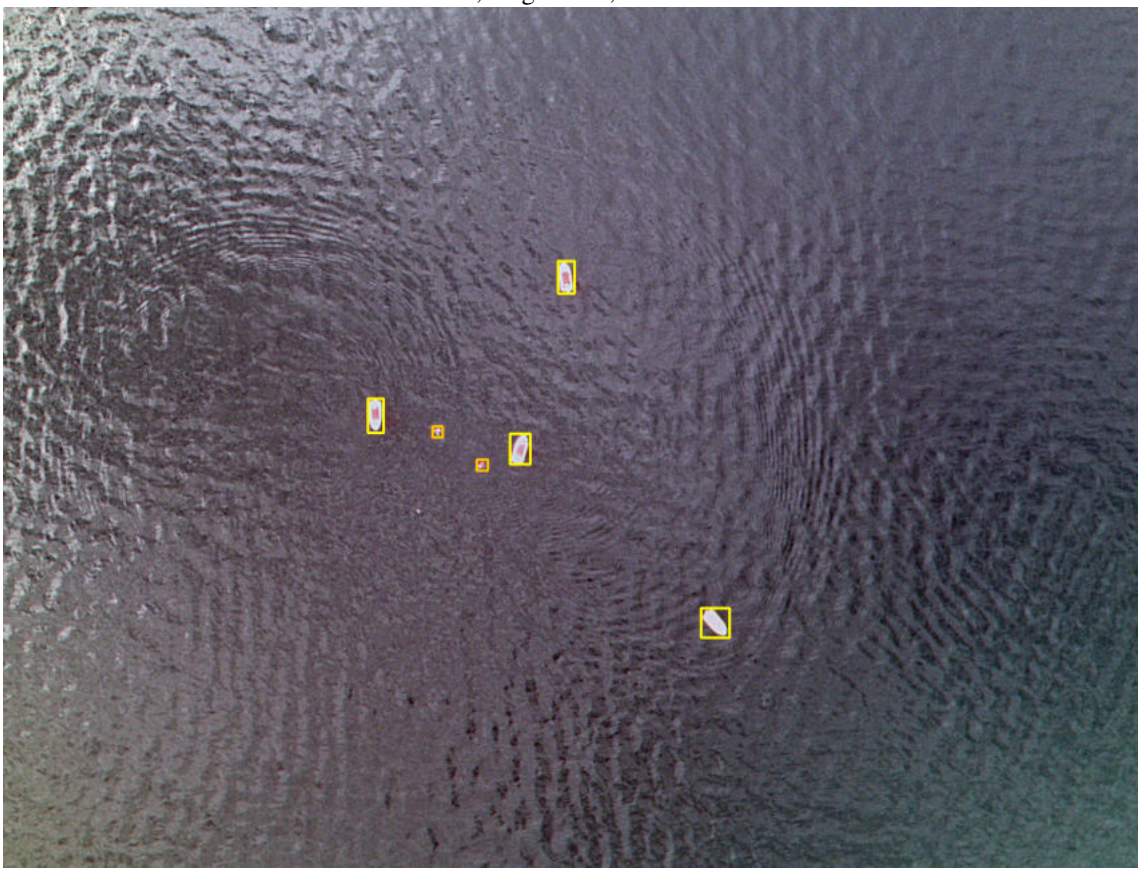
Altitude: 127 m; Angle: 77°; Camera: L1D-20C

Altitude: 10 m; Angle: 11°; Camera: L1D-20C

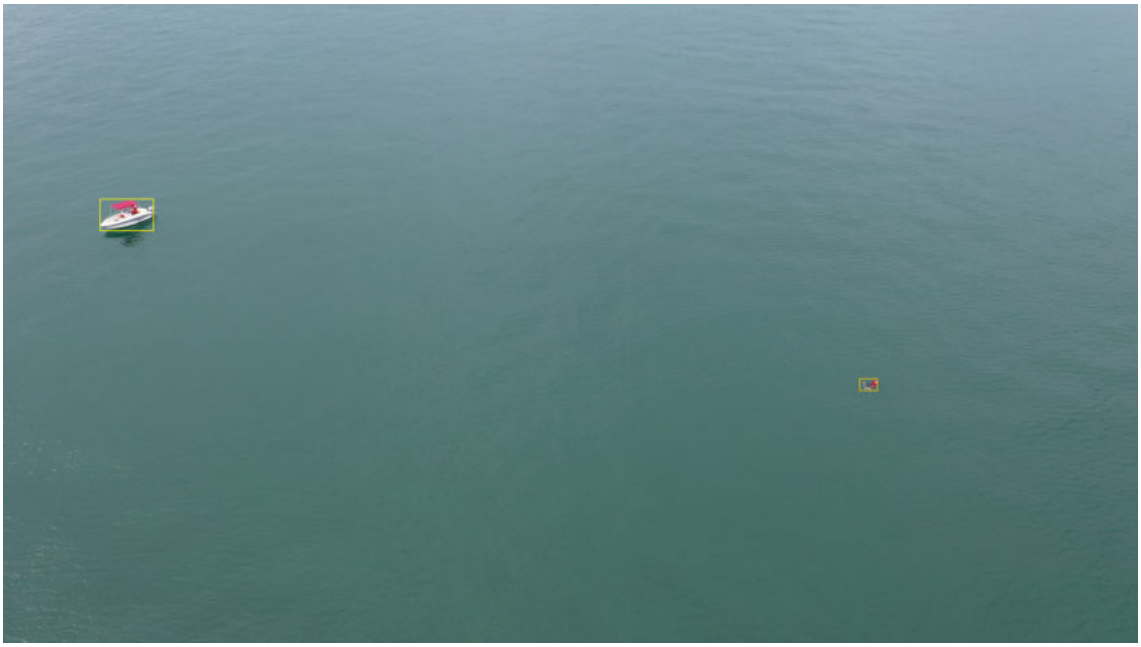


Altitude: 118 m; Angle: 90°; Camera: UMC-R10C

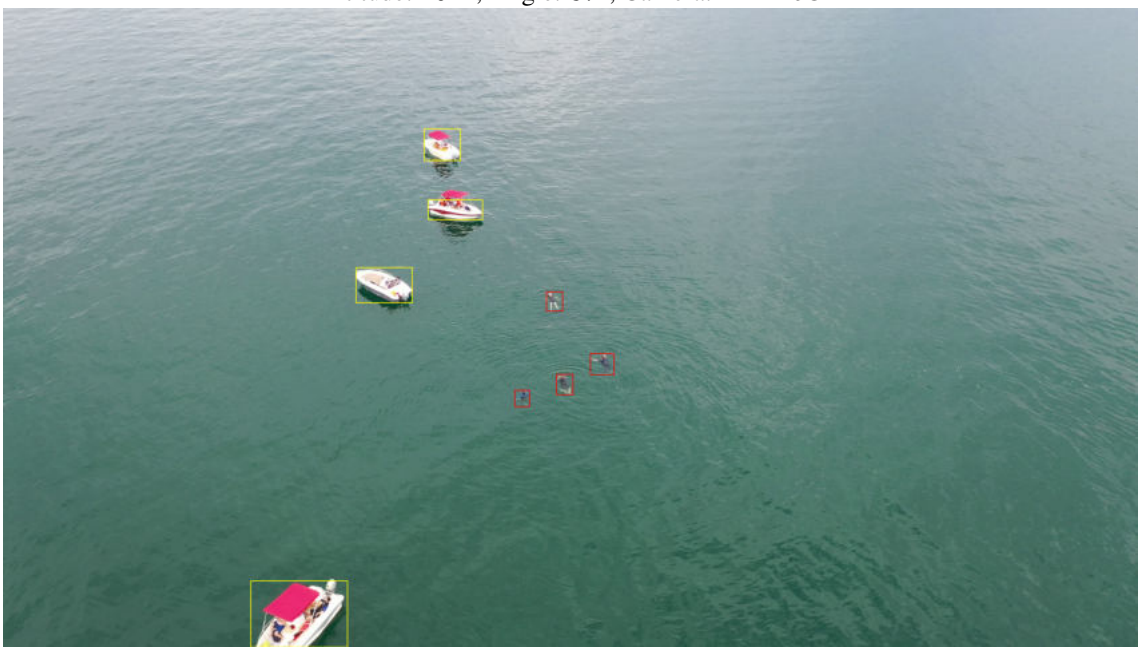
Altitude: 38 m; Angle: 39°; Camera: L1D-20C



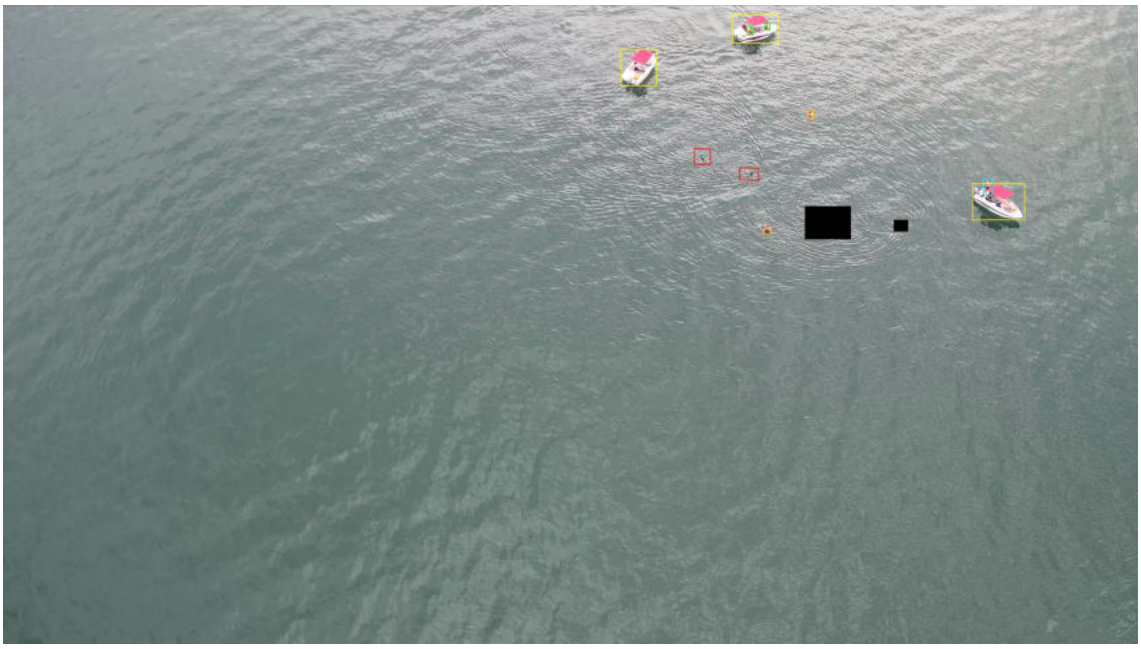
Altitude: 226 m; Angle: 90°; Camera: RedEdge-MX (RGB channels)



Altitude: 40 m; Angle: 37°; Camera: L1D-20C

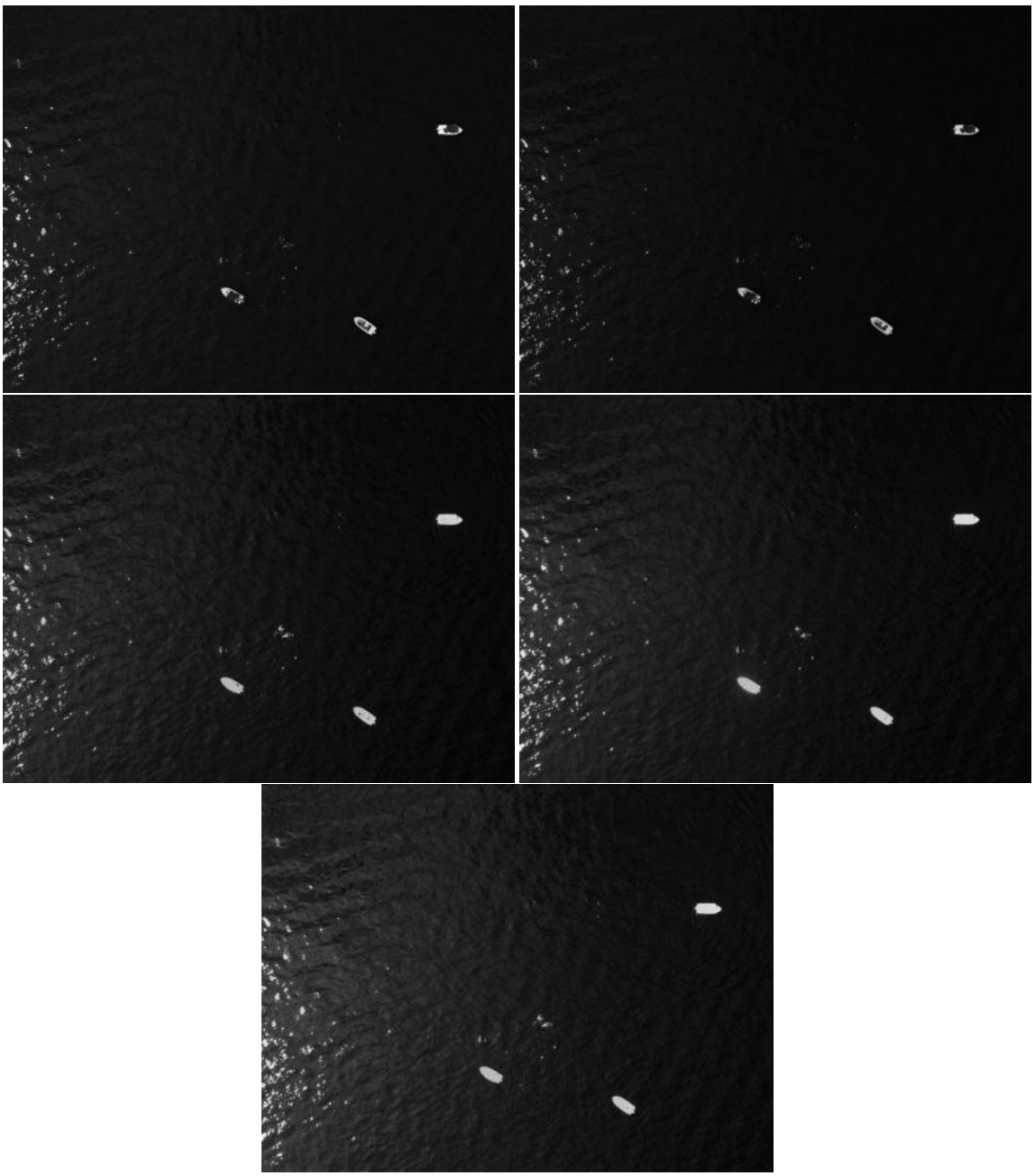


Altitude: 30 m; Angle: 31°; Camera: L1D-20C

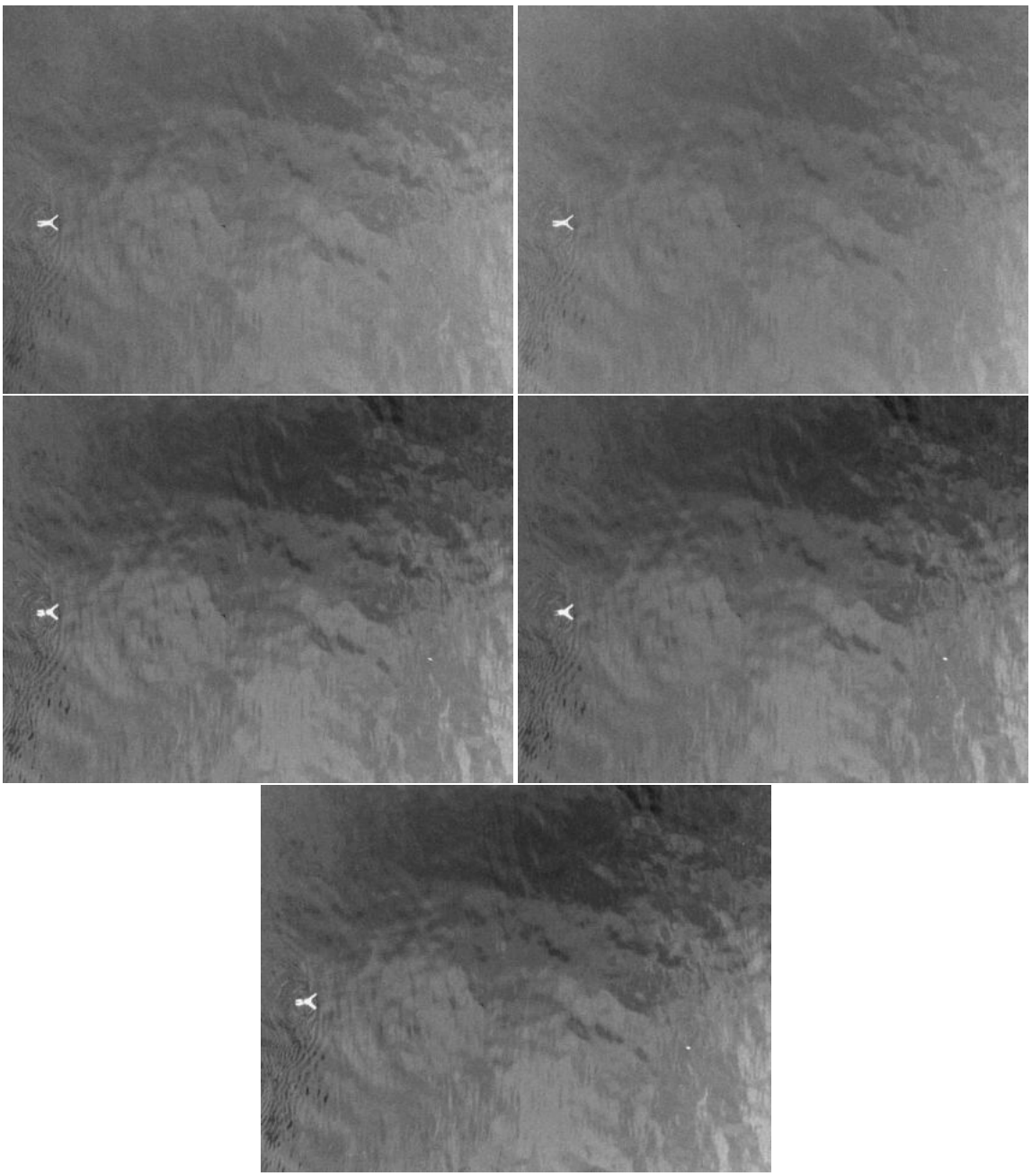


Altitude: 61 m; Angle: 41°; Camera: Zenmuse XT2

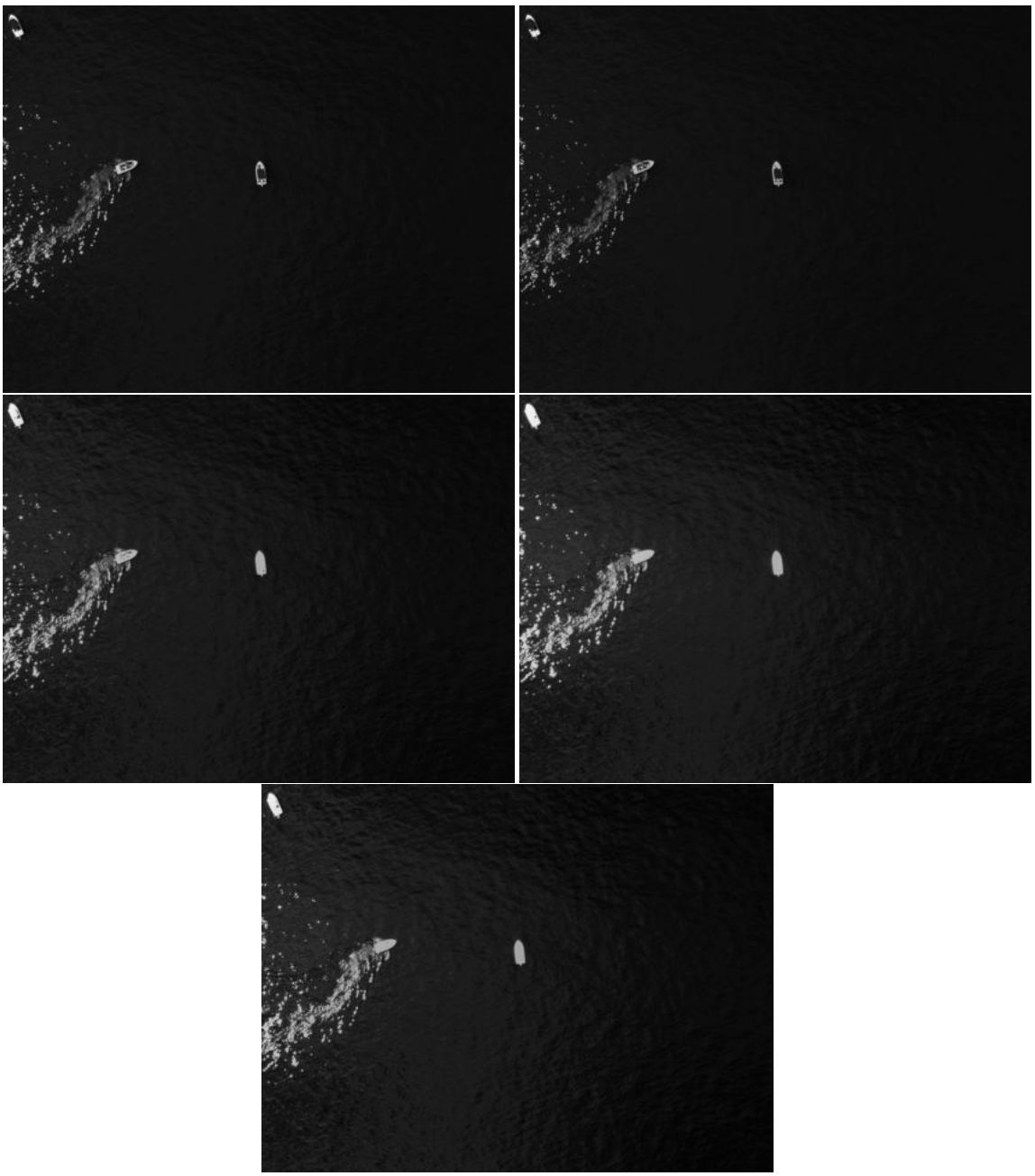
Altitude: 170 m; Angle: 90°; Camera: UMC-R10C



Multispectral image with the channels: blue (475 nm), green (560 nm), red (668 nm), red-edge (717 nm) and near-infrared (842 nm). [Altitude: 238 m; Angle: 90°; Time: 12:59:39]



Multispectral image with the channels: blue (475 nm), green (560 nm), red (668 nm), red-edge (717 nm) and near-infrared (842 nm). [Altitude: 54 m; Angle: 90°; Time: 12:48:15]



Multispectral image with the channels: blue (475 nm), green (560 nm), red (668 nm), red-edge (717 nm) and near-infrared (842 nm). [Altitude: 238 m; Angle: 90°; Time: 12:57:55]