A Deep-Learning Approach for Visual Detection of an AUV Docking Station

Faraz Ahmad[1,2], Tom Creutz[1], Christian Ernst Siegfried Koch[1], Bilal Wehbe[1]
[1]DFKI - Robotics Innovation Center
Germany, 28359 Bremen
Email: [1]{firstname.surname}@dfki.de, [2]faraz7321@gmail.com

***Abstract*—Autonomous docking for underwater vehicles, especially locating the docking station, presents significant challenges for deploying sub-sea resident AUVs in exploration and monitoring tasks. To extend a fiducial marker-based docking station detection we propose to use the state-of-the-art object detection deep learning models, specifically YOLOv8 in various sizes. We assess the robustness of these models in detecting docking stations by training different model sizes under various configurations on a dataset collected at the DFKI test basin in Bremen, Germany. To show and improve their performances in a real-world under-ice scenario we utilize a previously recorded dataset from Torneträsk lake in Abisko, Sweden [1]. The performance of these models is then compared to the previously used fiducial marker-based docking station detection. Our results show that a combination of both the classical detection method with one of the trained YOLOv8 models improves the detection performance significantly.**

***Index Terms*—AUV, autonomous docking, under-ice exploration, object detection, deep learning**

I. Introduction

Marine robotics are revolutionizing underwater exploration, industry and research, providing innovative solutions for environmental monitoring and offshore operations, while rapidly evolving to meet the diverse challenges of the marine world. The exploration and understanding of underwater environments have long posed significant challenges where autonomous underwater vehicles (AUVs) have emerged with key importance.

For extended duration missions, resident AUVs are required [2], which can operate over long periods without human intervention, making them ideal for sustained monitoring and exploration tasks. Their ability to dock, recharge, and transmit data autonomously allows for continuous operation in remote or harsh environments. Many areas of the ocean are difficult or dangerous for humans to reach, such as deep-sea trenches or under-ice regions. AUVs have the potential to safely operate in these conditions, providing valuable data that would otherwise be inaccessible. Additionally they can be equipped with a variety of sensors and instruments to collect high-quality data on oceanographic conditions, marine life, and underwater topography. This data is crucial for scientific research and environmental monitoring.

This work was funded by the Federal Ministry for Economic Affairs and Climate Action (BMWK), under the project the EurEx-LUNa (grant no. 50NA2002).

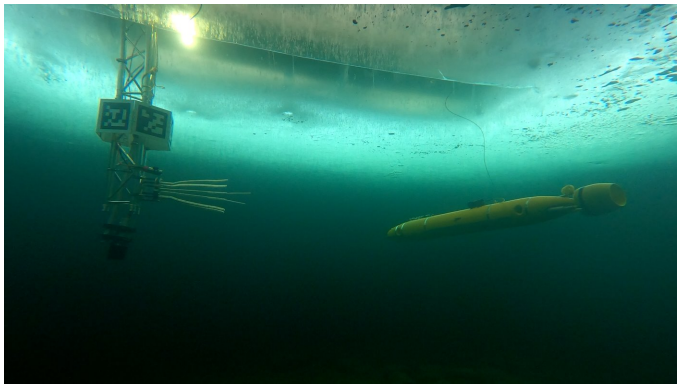


Fig. 1: Exploration AUV DeepLeng [5] approaching the docking station.

A specific field of application for resident AUVs is in under-ice environments [1], both on the terrestrial and extra-terrestrial settings. Over the past two decades, there has been an increasing interest in autonomous exploration beneath ice, focused on ocean monitoring, and climate research but also in searching for extra-terrestrial life beneath the icy surfaces of moons [3]. In such environments, the underwater docking process [4] is a critical aspect as it allows the AUV to return for charging, data transmission, and maintenance, but also increasing their efficiency and capability in underwater exploration. This paper will delve into the enhancement of the detection capabilities of AUVs during the autonomous docking process, a vital aspect for their continuous operation in challenging environments.

The primary goal of this research is the application of a state-of-the-art deep learning model and perform a thorough comparative analysis of its performance in recognizing an underwater docking station as shown in Figure 2, and fine-tuning the model to adapt to various visibility conditions. In environments with low visibility and poor lighting conditions, the fiducial Apriltag markers that are also visible in Figure 2 are not detectable by the Apriltag detector. Applying deep learning to this should help to improve these limitations. The learned model will be evaluated using the recorded datasets. In addition the unification of the existing docking station detection by Apriltag detection with the trained deep learning models is shown.

We aim to recognize a single, predefined static underwater docking station, providing a consistent reference across the different environments such as a natural lake [1] and a controlled test basin at DFKI in Bremen, Germany [6]. Comparison of performances of the deep learning models on distinct compositions of various recorded underwater datasets is done and different pre-processing steps are applied. As part of this process different model configurations are trained and evaluated to show there difference of performance and ability to generalize on the given environments.

II. Related Work

Perception is still a major challenge for AUVs operating in underwater environments [7]. Despite the impact of the challenging visual conditions such as lighting and turbidity, visual cameras are widely used for various object detection tasks. Most research towards object detection in underwater environment focuses on life objects such as coral reefs, various kind of living things but also artificial structures such as pipelines [8]–[18]. Within these research articles various methods and model architectures were used, such as various You Only Look One (YOLO) models [8], [11]–[17], ResNet [10], (Faster) RCNN [8], [11], [18], [19] or other model architectures [9]. There are also public datasets available as part of the Roboflow 100 dataset [20].

Most of these related research articles used various YOLO models [21] for object detection with great results in different domains. Over the years various improvements were made to the model architecture yielding both detection accuracy improvements and training efficiency increase. [22]–[24].

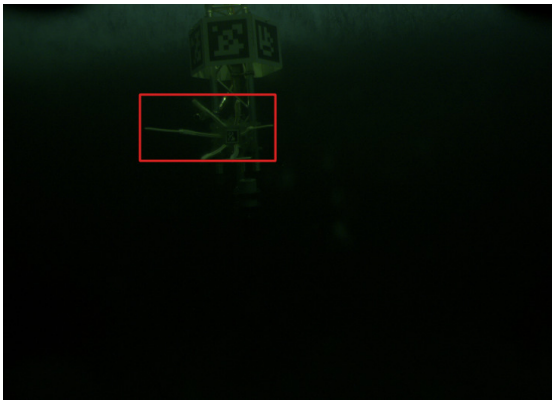
Another widely researched field in using object detection in underwater environments is in the usage of AUVs as sub-sea residents. The challenge for this kind of applications is the necessity to have a docking station which is used to recharge the AUV between different missions. Some research relies on using passive and active artificial fiducial markers on the docking station for detection and localization [4], [25]–[28]. While these methods already work reliably under good visual conditions, it is still difficult to detect a docking station under bad visual conditions.

III. Methodology

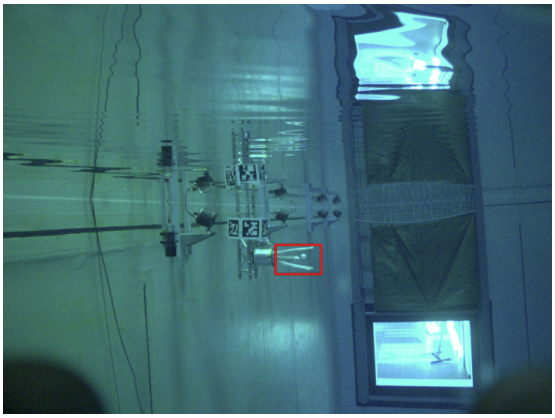
This section outlines the datasets collection preparation, model configuration, and experimental setup used for the detection capabilities of AUV docking station.

A. Datasets

In this paper, a pre-recorded under-ice dataset from Lake Torneträsk in Abisko, Sweden [1], [29] is utilized in addition to two datasets recorded in the basin at DFKI in Bremen, Germany [6]. The basin dataset was recorded from two separate sessions using the AUV Deepleng [5]. The recording contains images of the docking station at various angles, distances, and light conditions. The collected images are processed and organized to ensure consistency, with each image correctly labeled with a single class named ”docking_cone”.



(a)



(b)

Fig. 2: Images showing docking stations with bounding box as labels in Lake Torneträsk (a) and the test basin at DFKI facilities (b).

The datasets contain 1,650 images for the Abisko field trials dataset, and 2,090 in the DFKI basin dataset both recorded with a resolution of 2048 $\times$ 1536 pixels. The tool CVAT [30] was used labeling that images and for training the base model training, we used the DFKI basin dataset, with a split of 1600 images for training and 490 images for validation, from now on named dataset A. For transfer learning on the Abisko lake, we chose not to use the complete Abisko Field trials dataset but rather only utilize the images that were recorded from the start of the mission during undocking (79 images) and for validation, all of the frames during return and docking at the end of the mission (532 total images). We name this dataset B. In addition to that, during training of the transfer learning models, we handpicked 13 images from this larger validation set to validate during training process as there are lots of almost identical images in the bigger validation set. The left out images were only recorded before actually starting the exploration mission during the field trials. We chose to dismiss them for our final results due to their repetitiveness.

B. Model Parametrization

Building upon the decision to employ the YOLO framework for object detection, this paper will specifically utilize

YOLOv8, the latest iteration available when the research was started. It was developed and currently maintained by Ultralytics [23]. For training of our models we utilize the pre-trained weights on the COCO [31] dataset, rather than training from scratch.

1) Fine-Tuning and Model Development: Fine-tuning involves using pre-trained weights from an existing model and making small adjustments to enhance performance on a specific task or dataset. Unlike classical training, which starts from scratch, fine-tuning builds on a model that already has learned knowledge. This approach saves time and computational resources and is especially suitable when utilizing smaller datasets.

2) Hyper-parameter Tuning: Hyper-parameters are the high-level, structural configuration settings used to control the training process of the model. These are set before the training begins and remain fixed during the training process, unlike model parameters which are learned from the training data, such as weights in a neural network. They can have a significant impact on the model's performance and can affect the speed, efficiency, and accuracy.

To achieve the best results in training, hyper-parameters of the model needs to be configured and tuned according to the dataset. These parameters are frequently changed and updated during the tuning phase where multiple training sessions are executed. Some common hyper-parameters include learning rate, batch size, epochs, optimizer, and loss function but also data augmentation parameters.

3) YOLOv8 Training: As Ultralytics [23] provides multiple variants, four of these variants, *YOLOv8n*, *YOLOv8s*, *YOLOv8m*, and *YOLOv8l* are trained with their default hyper-parameters to observe how the model size impacts the performance on the custom dataset. *YOLOv8x* is excluded from consideration due to its substantial size and complexity relative to the available dataset. With a model size of 68.2M parameters and 257.8B FLOPs, *YOLOv8x* is significantly larger and computationally more demanding than the other variants. Given that our training dataset B contains only 2090 images in total, it would be too large for the available training data, potentially leading to over-fitting without a corresponding significant improvement in Average Precision.

4) Data Augmentation: Image augmentation can improve the robustness and performance of the models by introducing variability into the training data, helping the model generalize better to lake images. As it can be observed, the lighting conditions in the lake (Figure 2a) are much darker and monotonous as compared to the one from the basin (Figure 2b). Performing random jitters/adjustments to Hue, Saturation, and Value (HSV) channels of training images set could mimic the environment of under-ice lake. Figure 3 shows a sample image which is generated after applying random HSV values to the input image.

5) Transfer Learning: For fine-tuning our models both on the DFKI basin data and especially on the Abisko lake data, we use the principle of Transfer Learning [32]. By using pre-trained models (on the COCO dataset and on the DFKI basin dataset respectively) and freezing the first 10 layers of the model, we aim to transfer already learned feature detection to our specific domain. In addition to that we try to extend our limited datasets by utilizing data augmentation as mentioned in section III-B4.

C. Performance and Evaluation Metrics

Various metrics are used further in the paper to assess the performance of the deep learning models. These metrics include Precision and Recall, Average Precision (AP) and F1 Score. During our evaluation we focus mainly on Average Precision (*mAP50* and *mAP50-95*) to evaluate how well the model places and scales the bounding boxes and the Recall to assess its detection rate.

IV. Experiments and Evaluation

A. Experiment Setup

1) Description of the AUV: The AUV DeepLeng [5] is used to collect image data. It is equipped with several perception-related sensors and for this research particularly, Basler acA2040-35gc as a docking camera is utilized. Further information about the AUV can be found at [5].

B. Comparison of Different Model Configurations

As previously discussed, four variants of YOLOv8 are selected and trained. These models are trained and validated on dataset A with different configurations applied such as default hyperparameter setting from Ultralytics, freezing the backbone layers or applying data augmentation. An overview of the results is given in table I. For this comparison we chose a common batch size of 16.

In the baseline model configuration, no layers are frozen, and no data augmentation is applied, establishing the basic performance of the YOLOv8 variants (first 4 lines of table I). This provides a clear benchmark for understanding the impact of further configurations and modifications. Among these baseline models, the *YOLOv8m* model has the highest *mAP50* of 0.837.

After baseline model training, the effects of freezing the initial layers of the model are observed. In our case, the 'backbone' layers are frozen and only the 'Detect' module of the model is trained [33]. The data indicates that freezing layers can yield improvements in its performance. The *YOLOv8l* model with frozen backbone achieved an *mAP50* of 0.879 from 0.775, outperforming its baseline version. This suggests that freezing the backbone layers can generally improve docking station detection with more accuracy and precision but across the other 3 model sizes we observe a degradation across all metrics.

In the next steps, image augmentation is applied. It is noted that there is a profound impact on model performance, having significant improvements across all metrics. *YOLOv8s* with data augmentation and 10 frozen layers achieved an *mAP50* of 0.956 and high F1 score which represents a substantial improvement over the baseline or frozen layers version of itself.

Model	Batch Size	Frozen Layers	Data Augmentation	*mAP50*	*mAP50-95*	Precision	Recall	F1
YOLOv8n	16	0	NO	0.805	0.412	0.969	0.705	0.816
YOLOv8s	16	0	NO	0.59	0.296	0.999	0.585	0.738
YOLOv8m	16	0	NO	0.837	0.422	1	0.775	0.873
YOLOv8l	16	0	NO	0.775	0.46	0.998	0.656	0.792
YOLOv8n	16	10	NO	0.658	0.378	0.926	0.55	0.69
YOLOv8s	16	10	NO	0.531	0.282	0.988	0.44	0.61
YOLOv8m	16	10	NO	0.696	0.333	0.995	0.598	0.747
YOLOv8l	16	10	NO	0.879	0.509	0.947	0.832	0.662
YOLOv8n	16	0	YES	0.929	0.674	0.964	0.902	0.932
YOLOv8s	16	0	YES	0.792	0.448	1	0.632	0.774
YOLOv8m	16	0	YES	0.869	0.548	0.991	0.721	0.835
YOLOv8l	16	0	YES	0.894	0.592	0.979	0.838	0.903
YOLOv8n	16	10	YES	0.899	0.561	0.970	0.824	0.891
YOLOv8s	16	10	YES	0.956	0.721	0.977	0.957	0.967
YOLOv8m	16	10	YES	0.817	0.553	0.995	0.659	0.793
YOLOv8l	16	10	YES	0.922	0.594	0.985	0.851	0.913
YOLOv8m	8	10	YES	0.897	0.587	0.991	0.769	0.865

TABLE I: Top baseline models from each variant and config.

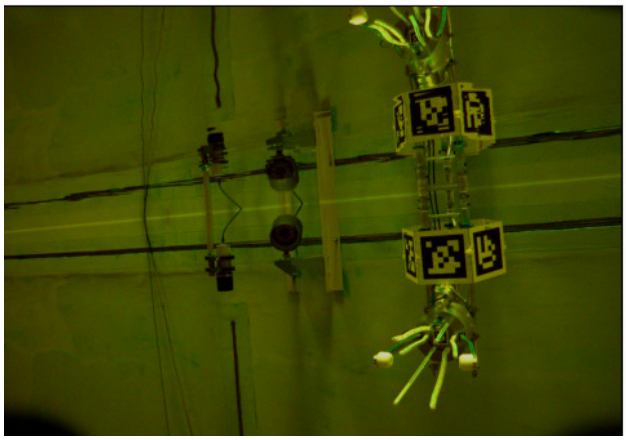


Fig. 3: Image generated with random HSV applied.

For further evaluation and transfer learning on real-world lake images, we chose the best performing model based on Recall and *mAP50* of each model size to go forward with. During our evaluation with different model sizes and training configurations, we also tested the impact of different batch sizes ranging from 8 to 32. While the best models of size *n, s* and *l* were all trained with a batch size of 16, we found that a batch size of 8 with data augmentation and freezing the bacbkone worked best for the *YOLOv8m* model. With a *mAP50* of 0.897, Recall of 0.769 and F1 score of 0.865 it outperformed the version with a batch size of 16 significantly. The results of this particular model were added to Table I and we marked the models in bold.

C. Evaluation on Real-World Lake Image Data

After training and validating the four different models on recorded test images from the basin, they were validated on real-world image data from dataset B which was collected during a field trip to the lake Torneträsk near Abisko, Sweden [1]. To recall, we selected images that contained the docking station from this dataset [29] with an image index greater than 5122, which corresponds to the AUV returning from a mission to the docking station, as the validation data set. The models were validated on a total of 532 images and the default validation parameters. In terms of Recall the *YOLOv8n* model performs best on validation dataset B with 0.613 and a *mAP50* performance of 0.483. The most accurate bounding boxes were predicted by the *YOLOv8l* model with a *mAP50* value of 0.619. While these performance metrics are worse compared to validation with dataset A shown in table I, the models are already able to generalize on the real-world dataset B from the Torneträsk lake in Abisko.

D. Transfer Learning (TL) for real-world scenario

To further improve the docking station detection performance of the chosen models we composed dataset B of only 79 images for training which were recorded during the undocking process in the dataset [29]. It contains only positive docking station examples with mostly close-up frontal view of the docking station. As validation data during the training process we handpicked a set of only 13 images from the 532 previously used images during the return to the docking station. Using this data, the pre-trained models from IV-B were trained and compared in a similar fashion, using different frozen layers, data augmentation and batch size configurations. In table II only the configuration and metrics of the best performing transfer learning models are shown. All models perform almost identical on the selected 13 images. The *YOLOv8s* model stands out with a perfect F1 score on the selected validation data, while the *YOLOv8m* model achieves the most accurate detection with a *mAP50* and *mAP50-95* value of 0.99 and 0.465 respectively. The high precision and recall values in this case are observed due to the limited validation set of only 13 images which was used during the training process.

1) Comparison between fine-tuned and base models: For better comparison, we chose to evaluate the transfer learning models, trained on the selected lake images (dataset B), to the baseline models, which were only trained on the test basin dataset A. For validation we took the bigger 532 images validation dataset from B. The comparison is shown in Figure

Model	Batch Size	Frozen Layers	Data Augmentation	mAp50	*mAP50-95*	Precision	Recall	F1
YOLOv8n	2	10	Yes	0.906	0.44	1	0.846	0.916
YOLOv8s	8	10	Yes	0.909	0.463	1	1	1
YOLOv8m	8	10	Yes	0.99	0.465	1	0.971	0.9851
YOLOv8l	8	10	Yes	0.925	0.463	0.99	0.846	0.914

TABLE II: Model validation on lake images during return to docking station after fine-tuning the model.

4. The baseline models are visualized in blue, the transfer learning models in orange.

Most of the fine-tuned models perform better than their baseline models across all metrics. We observe an improvement in bounding box precision with *mAP50* values being slightly better across all models. Still the gains are only marginal, except for the *YOLOv8l* model, where we can see an increase of performance of roughly 29%. In terms of recall there is actually a degradation for the *YOLOv8n* and *YOLOv8l* models. Still due to the quite significant increase in terms of precision the overall F1 score is increased across all models. Noteworthy is that in this experiment due to the default setting there can be multiple detections per frame that are not being suppressed by the default Non-Maxima Supression (NMS) setting of an default IoU of 0.6. To robustify the detections for the upcoming evaluations, we therefore limit the maximum number of detections to 1 per frame as we can assume to only see one single docking station per image frame. The single detection is chosen by the highest confidence value. As visualized in Figure 4 this detection parameter tuning of limiting the detection count increases the performance across all models significantly.

E. Comparison to traditional AprilTag Detection and combination of both methods

To show the contribution of the trained models to detecting the docking station when returning back to it after a mission, we run AprilTag detection on all images of the validation set to compare detection performance. With the Apriltag detections being very robust against false positives, we assume that if one detection is found in the image, it successfully detected the docking station. We then calculate a Recall metric which represents the normalized count of successful detections within the validation data set. For the Apriltag [34] detector all the default parameters are used except *quad_decimate* is set to 6.0. With these parameters, we achieve the most observations of 152 images having a positive detection, which yields a recall of 0.286 on the utilized validation set of 532 images.

1) Combination of Apriltag and Transfer Learning Models: To achieve the best detection results we propose to combine the Apriltag detector with the fine-tuned models presented in section IV-D. First and foremost for this comparison we set the used detection confidence threshold from 0.001 which was used in the previous section IV-D1 to 0.25. We retrieve this value from the output confidence matrix that is generated during the validation process of the model using the ultralytics library [23]. In Figure 4 the precision and recall values are calculated at the confidence value of 0.001. For a confidence value of 0.25 we observe no false positive detections for any model but a decreased recall of 0.58, 0.49, 0.57 and 0.54 across the different model sizes from *n* to *l*. For the combination of both detection methods we define a positive detection if atleast the model or the Apriltag detector detects the docking station in a given image. For validation, we use the validation images consisting of 532 images from returning to the docking station after a mission. The recall of this unification shows that the *YOLOv8n* and *YOLOv8m* models perform best with a total count of detections of 386 or a recall of 0.726. The other models perform similarly with the *YOLOv8s* and *YOLOv8l* models having a recall of 0.694 and 0.724 respectively. While overall we see an improvement compared to the previously observed recall using the 0.25 confidence threshold, this result is still on-par with the results from solely using the models with a confidence threshold with 0.001. But using this lower confidence threshold would then again result in potential false-positive detections that can hinder performance.

F. Model Performance on hand-picked image samples

The downside of the used validation data set is that it only contains positive samples that actually contain the docking station. In addition to that, it is hard to quantify its performance on images with particular harsh conditions, such as dark lighting, longer distances, or unique viewing angles. To show the performance of the models we selected a handful of images that were not part of the previously used validation data set. In figures 5a and 5b we show examples of a typical scenario when approaching the docking station from the front. All but the *YOLOv8l* models were able to detect the docking station in these two handpicked samples. The *YOLOv8l* only detected the docking station in one of the two basic scenario images. In figures 5c and 5d we show remarkable detection results. Our models based on *YOLOv8m* and *YOLOv8l* are able to successfully detect the docking station while barely being visible at the edge of the image in figure 5c. Only our *YOLOv8s* model was able to detect the docking station in Figure 5d which shows robustness even to very sub-optimal lighting conditions. In figures 5e-5h we show some odd false positives that occurred across various models. Besides of the shown examples, the models were also evaluated against several examples from both datasets not containing the docking station and even some other structures in the basin. All of the models correctly did not detect the docking station in any of those examples.

V. Discussion

In this work we evaluate a selection of YOLOv8 models trained with varying training configurations and model sizes in the context of detecting an underwater docking station for an AUV. We utilize two datasets, one recorded in the test basin at DFKI, Bremen [6] and one recorded on a field trip to the Torneträsk lake in Abisko, Sweden [1], [29]. By comparing the application of data augmentation during the training process and freezing of the first ten layers of the models we selected the best performing model of each size according to their mAP and Recall on the test basin validation set. The assumption of freezing the backbone layers of the pre-trained model hold for 3 out 4 models. Also we can clearly see the benefits of augmenting the training data set to virtually generate more training samples. We went forward to evaluate these selected models on a validation set from the Abisko field trials to show its generalization capabilities to another environment, a frozen lake with far worse lighting and slightly worse visual conditions. We then applied transfer learning by fine-tuning the model with a limited dataset of only 79 image samples (but using data augmentation during the training process) to increase its Recall performance by up to 86% to a maximum of 0.549 for the YOLOv8n model. In the end we showed its impact when combining it with the previously used fiducial marker detection (Apriltags) for docking station detection. Using our validation dataset we were able to boost the number of successful docking station detectios from 152 images ($Recall = 0.286$) to 378 images ($Recall = 0.72$) for the best performing model (*YOLOv8n*). In addition to that, all of the models were able to detect the docking station even in very harsh visual conditions which should lead to improved homing performance of the AUV when adding this method to the docking process. Nevertheless the actual deployment and evaluation of such models in the real-world scenario still remains future work.

A. Outlook

To continue work on the proposed methods we aim to refine various processes of the model development, data processing and deployment processes. Most of the work was conducted during the thesis work of the main author. When composing the results, YOLOv8 was still the best performing model selection of its kind. By now the newest version YOLOv10 is available and even comes with improved detection performance in addition to training and inference efficiency due to decreased parameter count [24]. In addition to that there are also various different object detection methods out there such as Faster R-CNN [19] or Transformer-based object detection such as DETR [35].

In addition to that we relied heavily on the built-in data augmentation and pre-processing functionalities, that the Ultralytics YOLOv8 library offers [23], to train our models. In the future it could be beneficial to apply customly implementation pre-processing and data augmentation functionalities that could then also possibly be used during live deployment and for inference.

Last but not least we want to actually deploy these models on the AUV to extend our existing docking algorithm that was proposed in [4]. For this the AUV Deepleng needs to be extended with an onboard computer that is able to infer from images in real-time. Benchmarks suggest that even the YOLOv8l model will run in real-time ($>$ 30FPS) on a 32GB VRAM NVIDIA Jetson module [36]. Our best performing model from this work, which is the YOLOv8n model would even run with over 60FPS with FP32 precision on a 8GB VRAM device.

In addition for future work it will be very beneficial to not only detect the docking station but also estimate its pose using a monocular image input or a combination of images and sonar images.

References

[1] M. Hildebrandt, T. Creutz, B. Wehbe, M. Wirtz, and M. Zipper, "Under-Ice Field tests with an AUV in Abisko/Torneträsk," in *OCEANS 2022-Hampton Roads*. IEEE, 2022.

[2] *FlatFish Resident AUV: Leading the Autonomy Era for Subsea Oil and Gas Operations*, ser. OTC Offshore Technology Conference, vol. Day 4 Thu, May 03, 2018, 04 2018. [Online]. Available: https://doi.org/10.4043/28881-MS

[3] M. Hildebrandt, J. Albiez, M. Fritsche, J. Hilljegerdes, P. Kloss, M. Wirtz, and F. Kirchner, "Design of an autonomous under-ice exploration system," in *2013 OCEANS - San Diego*, 2013, pp. 1–6.

[4] T. Creutz, B. Wehbe, S. Arnold, and M. Hildebrandt, "Towards robust autonomous underwater docking for long-term under-ice exploration," in *OCEANS 2023 - Limerick*, 2023, pp. 1–8.

[5] DFKI, "Exploration auv deepleng." [Online]. Available: https://robotik.dfki-bremen.de/en/research/robot-systems/deepleng

[6] G. R. C. for Artificial Intelligence GmbH, "Robotics innovation center dfki." [Online]. Available: https://robotik.dfki-bremen.de/en/research/research-facilities-labs/maritime-infrastructure

[7] L. Christensen, J. Fernández, M. Hildebrandt, C. Koch, and B. Wehbe, "Recent advances in ai for navigation and control of underwater robots," *Current Robotics Reports*, 08 2022.

[8] S. Sheth and D. J. Prajapati, "Recognition of underwater starfishes using deep learning," in *2022 Second International Conference on Next Generation Intelligent Systems (ICNGIS)*. IEEE, 2022, pp. 1–5.

[9] F. Peng, Z. Miao, F. Li, and Z. Li, "S-fpn: A shortcut feature pyramid network for sea cucumber detection in underwater images," *Expert Systems with Applications*, vol. 182, p. 115306, 2021.

[10] Z. Zhao, Y. Liu, X. Sun, J. Liu, X. Yang, and C. Zhou, "Composited fishnet: fish detection and species recognition from low-quality underwater videos," *IEEE Transactions on Image Processing*, vol. 30, pp. 4719–4734, 2021.

[11] G. M. Boris Gašparović, Jonatan Lerga and M. Ivašić-Kos, "Deep learning approach for objects detection in underwater pipeline images," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2146853, 2022. [Online]. Available: https://doi.org/10.1080/08839514.2022.2146853

[12] M. Hamzaoui, M. Ould-Elhassen Aoueileyine, L. Romdhani, and R. Bouallegue, "An improved deep learning model for underwater species recognition in aquaculture," *Fishes*, vol. 8, no. 10, 2023. [Online]. Available: https://www.mdpi.com/2410-3888/8/10/514

[13] H.-S. Jin, H. Cho, H. Jiafeng, J.-H. Lee, M.-J. Kim, S.-K. Jeong, D.-H. Ji, K. Joo, D. Jung, and H.-S. Choi, "Hovering control of uuv through underwater object detection based on deep learning," *Ocean Engineering*, vol. 253, p. 111321, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801822007132

[14] D. N. Venkatesh Alla, V. Bala Naga Jyothi, H. Venkataraman, and G. Ramadass, "Vision-based deep learning algorithm for underwater object detection and tracking," in *OCEANS 2022 - Chennai*, 2022, pp. 1–6.

[15] Y. Ouassine, Z. Jihad, N. Conruyt, P. Martin, B. Lionel, M. Kayal, C. Eric, and R. Vignes-Lebbe, "Automatic coral detection with yolo: A deep learning approach for efficient and accurate coral reef monitoring," 01 2024, pp. 170–177.

[16] H. Yang, P. Liu, Y. Hu, and J. Fu, "Research on underwater object recognition based on yolov3," *Microsystem Technologies*, vol. 27, no. 4, pp. 1837–1844, 2021.

[17] X. Zhao, X. Wang, and Z. Du, "Research on detection method for the leakage of underwater pipeline by yolov3," in *2020 IEEE international conference on mechatronics and automation (ICMA)*. IEEE, 2020, pp. 637–642.

[18] H. Huang, H. Zhou, X. Yang, L. Zhang, L. Qi, and A.-Y. Zang, "Faster r-cnn for marine organisms detection and recognition using data augmentation," *Neurocomputing*, vol. 337, pp. 372–384, 2019.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016. [Online]. Available: https://arxiv.org/abs/1506.01497

[20] F. Ciaglia, F. S. Zuppichini, P. Guerrie, M. McQuade, and J. Solawetz, "Roboflow 100: A rich, multi-domain object detection benchmark."

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[22] G. Jocher, "Ultralytics yolov5," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

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

[24] A. Wang, H. Chen, and L. Liu, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.

[25] N. Palomeras, G. Vallicrosa, A. Mallios, J. Bosch, E. Vidal, N. Hurtos, M. Carreras, and P. Ridao, "Auv homing and docking for remote operations," *Ocean Engineering*, vol. 154, pp. 106–120, 2018.

[26] K. N. Lwin, N. Mukada, M. Myint, D. Yamada, M. Minami, T. Matsuno, K. Saitou, and W. Godou, "Docking at pool and sea by using active marker in turbid and day/night environment," *Artificial Life and Robotics*, vol. 23, no. 3, pp. 409–419, 2018.

[27] T. Wang, Q. Zhao, and C. Yang, "Visual navigation and docking for a planar type auv docking and charging system," *Ocean Engineering*, vol. 224, p. 108744, 2021.

[28] T. L. Rannnestad, A. Waldum, and M. Ludvigsen, "Visual close-range navigation and docking of underwater vehicles," in *OCEANS 2023 - Limerick*, 2023, pp. 1–10.

[29] Hildebrandt, Marc and Wehbe, Bilal and Wirtz, Marius and Creutz, Tom and Zipper, Michael, "Eurex-LUNa Abisko Trials Lawnmower Vectoring." [Online]. Available: https://zenodo.org/record/7035132

[30] CVAT.ai Corporation, "Computer vision annotation tool (cvat)," 2023. [Online]. Available: https://github.com/opencv/cvat

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[32] M. Iman, H. R. Arabnia, and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technologies*, vol. 11, no. 2, p. 40, Mar. 2023. [Online]. Available: http://dx.doi.org/10.3390/technologies11020040

[33] R. King, "Yolov8 architecture." [Online]. Available: https://github.com/RangeKing

[34] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[35] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020. [Online]. Available: https://arxiv.org/abs/2005.12872

[36] Lakshan, "Yolov8 benchmarks on nvidia jetson modules." [Online]. Available: https://www.seeedstudio.com/blog/2023/03/30/yolov8-performance-benchmarks-on-nvidia-jetson-devices/

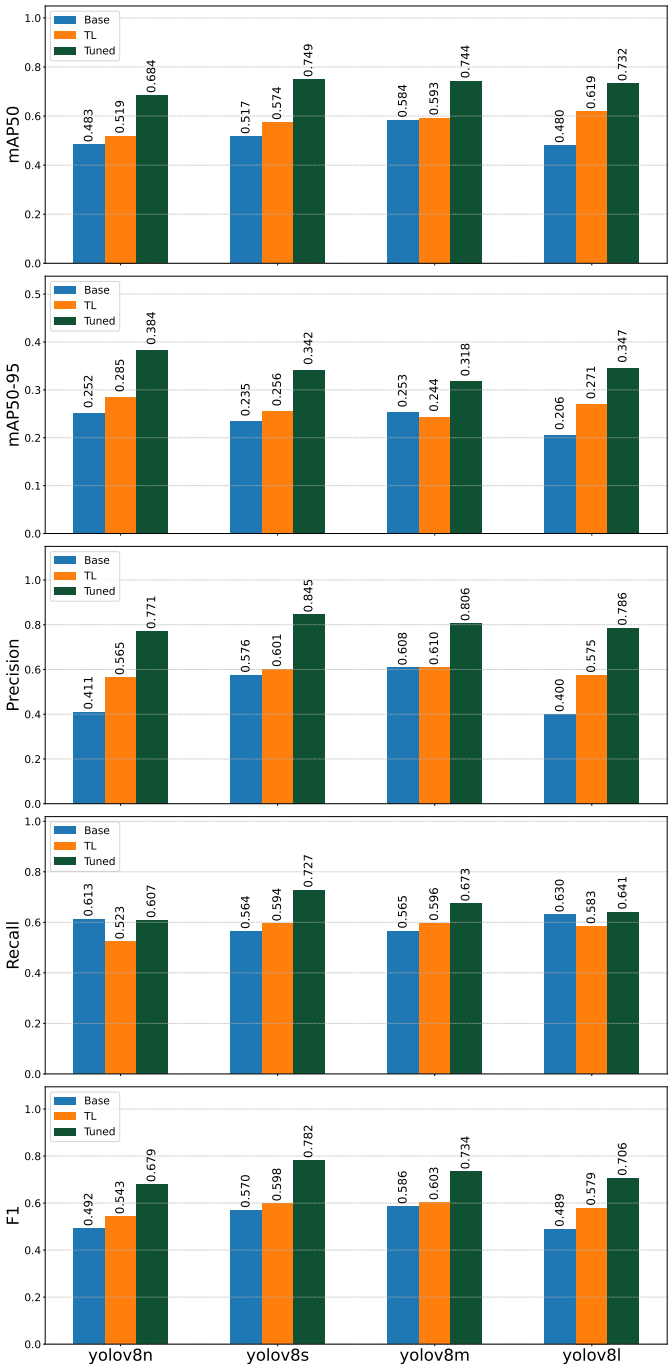
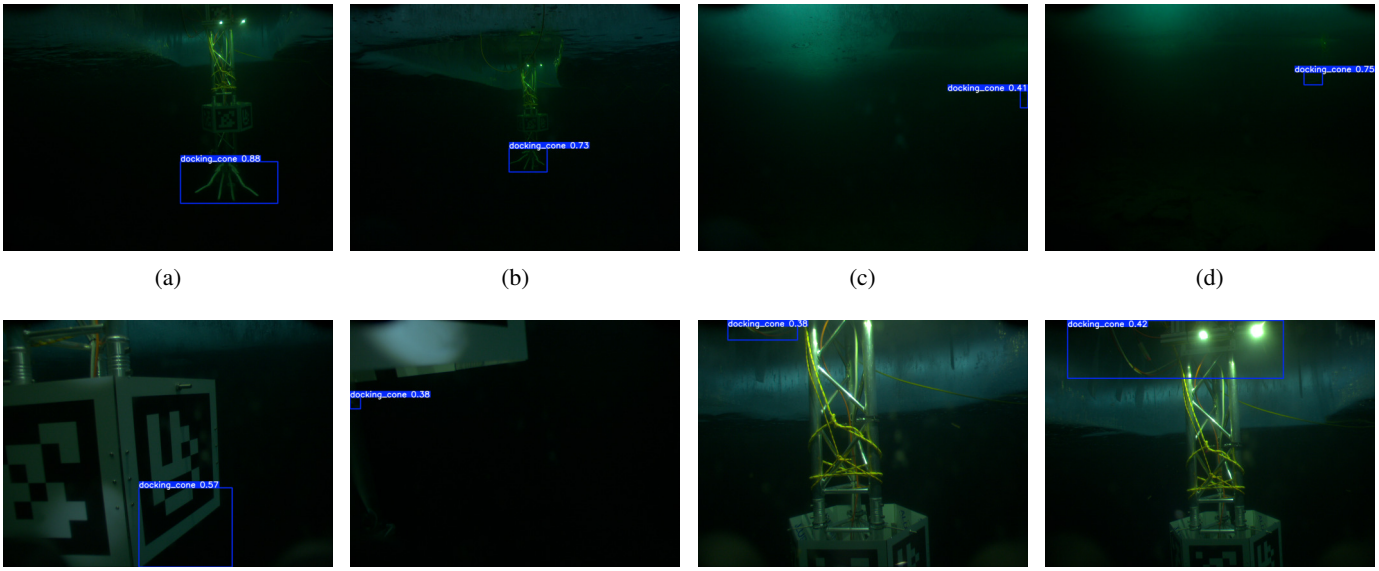


Fig. 4: Comparison of different metrics between the baseline models and the transfer learning (TL) models. The base models are visualized in blue, the transfer learned models are shown in orange. In addition the that the performance of each model with the detection count per image limited to 1 is shown in green.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 5: Exemplary detections of different models to show performance on selected image samples.