Following scuba divers with an autonomous underwater vehicle

Khalil Virji
Master of Science

School of Computer Science
McGill University
Montreal, Quebec, Canada

August 11, 2024

A thesis submitted to McGill University in partial
fulfillment of the requirements of the degree of
Master of Science

Abstract

In this thesis, we address the problem of following scuba divers with an autonomous underwater vehicle (AUV). Our primary objective is to enhance underwater human-robot collaboration by enabling AUVs to track and follow scuba divers robustly in the open ocean. We design a custom visual servoing system that directly controls the motion of our AUV using visual feedback from an onboard camera. To comprehensively address the perception and control challenges involved in autonomously following scuba divers, we include distinct modules for each component. For perception, we develop a vision-based system that integrates a state-of-the-art object detector with a multi-object tracker. This system reliably detects and tracks scuba divers in the image plane while handling challenges such as missing and false detections. In addition to accuracy, we evaluate our vision system using second-order temporal stability metrics, which are crucial in visual servoing where perception and control are tightly intertwined. For control, we explore both traditional and learning-based paradigms. We design Proportional-Integral-Derivative (PID) controllers to control our AUV solely using visual feedback and demonstrate their effectiveness through deployments in the open ocean.

To address the inherent limitations of PID control, we investigate the potential for a learning-based controller that can encode complex behaviors and autonomously adapt to changes in the system and environment. Through a series of experiments, we compare traditional PID controllers with learning-based approaches, providing valuable insights for real-world deployments. Our findings contribute not only to the specific application of scuba diver following but also to the broader field of visual servoing in marine robotics. Our framework can be easily generalized to a wide range of underwater tasks, advancing methods for underwater human-robot collaboration and autonomy.

Abrégé

Dans cette thèse, nous adressons le problème du suivi de plongeurs à l'aide de drones sous-marins autonomes (AUV). Notre objectif principal est d'améliorer la collaboration entre homme et robot en milieu sous-marin en permettant aux AUVs de détecter et suivre les plongeurs en eau libre. Nous concevons un système d'asservissement visuel qui contrôle directement les mouvements de notre AUV grâce au retour visuel d'une caméra à bord. Afin de complètement adresser les défis de perception et de contrôle qui s'impliquent dans le suivi en autonomie de plongeurs, notre solution inclut un module distinct pour chaque composant. Pour la perception, nous développons un système visuel qui intègre un détecteur d'objets de pointe avec un tracker multi-objet. Ce système détecte et traque les plongeurs dans le plan image tout en étant robustes aux défis tels que les détections manquantes et les fausses détections. En plus de la précision, nous évaluons notre système visuel au travers de métriques de stabilité temporelle du deuxième ordre, qui sont cruciaux au servoing visuel, où la perception et le contrôle sont étroitement liés. Pour le contrôle, nous explorons aussi bien les approches traditionnelles que les approches basées sur l'apprentissage. Nous développons des

contrôleurs Proportionnel-Intégral-Dérivés (PID) pour contrôler notre AUV à l'aide du retour visuel uniquement, et montrons leur efficacité au travers de déploiements en eau libres, dans l'océan. Afin de remédier aux limitations innées du contrôle PID, nous étudions aussi le potentiel d'un contrôleur basé sur l'apprentissage, qui est capable de comportements complexes et de s'adapter en autonomie aux changements du système et de l'environnement. Par une série d'expériences, nous comparons les contrôleurs traditionnels PID avec les contrôleurs basés sur l'apprentissage, fournissant des renseignements utiles aux fins du déploiement dans le monde réel. Nos résultats contribuent non seulement à l'application spécifique qu'est le suivi des plongeurs sous-marins, mais aussi plus largement au domaine de l'asservissement visuel en robotique marine. Notre système peut être facilement généralisé à une grande variété de tâches sous-marines, faisant progresser les méthodes pour la collaboration sous-marine entre robots et humains.

Contributions

This thesis presents a visual servoing framework for following scuba divers with an autonomous underwater vehicle. The author is a primary contributor to the following:

- Developing a lightweight vision module combining You Only Look Once version 7 (YOLOv7) for diver detection with Simple Online and Real-time Tracking (SORT) for diver tracking.
- Collecting and annotating a novel dataset for use in simulation experiments.
- Investigating the effect of SORT on detection accuracy and temporal stability when combined with YOLOv7.
- Proposing improvements to existing temporal stability metrics to better capture frame-to-frame stability, which is crucial in visual servoing applications.
- Designing and tuning Proportional-Integral-Derivative (PID) controllers to autonomously follow a scuba diver using visual feedback.

- Developing a spiral search mechanism to quickly recover scuba divers after losing visual contact.
- Conducting open ocean experiments to validate the feasibility and performance of the presented framework.
- Formulating scuba diver following as a Markov decision process (MDP) and utilizing reinforcement learning (RL) to learn a control policy.
- Analyzing the strengths and limitations of traditional versus learning-based control paradigms.
- Open-sourcing all code used in this research at github.com/khalilv/aqua_mrl.

Acknowledgments

To the many individuals who have supported me throughout my graduate studies, thank you. None of this work would have been possible without your continuous guidance, encouragement, and support.

To my family, thank you for your love and support throughout my studies at McGill University. I am blessed to have been raised in such a loving and principled environment, surrounded by a family that is always looking out for me.

To my supervisor, Gregory Dudek, thank you for your mentorship, encouragement, and wisdom. Our discussions have always left me inspired and full of innovative ideas, providing clear directions for my research. To Kaleem Siddiqi, a professor I respect and admire, thank you for igniting my passion for research and for giving me the opportunity to continue my studies at McGill.

To my friends and colleagues at the Mobile Robotics Laboratory, thank you. One thing I didn't fully grasp coming into graduate school was how I would be surrounded by such intelligent, resourceful, and kind researchers. It has been a pleasure to work, collaborate, joke, and enjoy lunch with you all.

A special thank you to my friend Faraz Lotfi, with whom I worked closely throughout my studies. Your guidance, knowledge, and patience have been instrumental in shaping my growth as a researcher.

To my friends outside of the lab, thank you for making my time in Montreal such a rewarding, fun, and memorable experience. I often wonder how I got so lucky to cross paths with such driven and unique individuals, but I am eternally grateful to have you all in my life.

Contents

List of Figures

List of Tables

1

Introduction

Scuba diving in the open ocean offers a unique way to observe and explore the marine world. Despite its appeal, limited visibility, strong currents, and the challenges of movement in all directions make scuba diving an inherently complex and risky activity. Standard diving protocols require all dives to be planned and executed in pairs of divers, emphasizing the importance of frequently monitoring one another and staying close for reliable communication and safety [77].

Staying together not only makes the diving experience more enjoyable, but it is also pivotal because underwater, divers are unable to verbally communicate. Instead, they rely on communicating via hand signals, which require close visual contact for both sending and receiving messages. Besides routine communication, divers need to stay together to identify and quickly

respond to emergencies, such as equipment malfunctions, air shortages, or injuries.

During dives, scuba divers are often required to navigate and survey along a predefined route. In these scenarios, maintaining awareness of their partner is key to avoid separation or getting lost. It is remarkably easy to become distracted or disoriented underwater, especially in unfamiliar regions. Introductory diver training rigorously emphasizes the importance of staying together during dives and quickly recovering from extended separation. Standard safety protocols dictate that losing sight of a partner for more than a minute requires divers to ascend to the surface to regroup, potentially ending the dive prematurely [77].

Underwater human-robot collaboration is a promising area of research within the robotics community. Given their advanced sensing capabilities, the idea of autonomous underwater vehicles (AUVs) working alongside scuba divers is encouraging for tasks like underwater mapping and oceanic monitoring. However, for AUVs to operate effectively as scuba diver companions, they must first be capable of tracking and following their fellow divers. In communication scenarios, an AUV must maintain close visual contact for hand signal or gesture recognition. In emergency situations, an AUV must be within range to identify the problem and provide assistance. During underwater navigation, an AUV must reliably follow a team to reach a workspace or inspection site.

While scuba divers can naturally monitor and follow each other through-

out a dive, devising an autonomous diver following system for an AUV requires careful design and consideration. In particular, diver following with an AUV is a task that combines both perception and control. First, the AUV must accurately perceive a diver and determine its relative location. Diver perception using sonar and by detecting flipper oscillations in the frequency domain are approaches that have been explored [25, 59, 80]; however, these have been vastly outnumbered by vision-based methods using an onboard camera [2, 14, 37, 83, 98]. Vision-based diver perception is popular because of its cost-effectiveness over alternative sensing mechanisms, its general applicability in a variety of underwater conditions and tasks, and its ability to integrate advanced computer vision techniques to extract high-dimensional information from images.

Second, for an AUV to autonomously follow a diver in the open ocean, it must incorporate a robust control system. Unlike diver perception mechanisms that generalize well for use on any AUV, control systems need to be carefully designed for the dynamics and intricacies of the particular platform. Proportional-Integral-Derivative (PID) controllers, which require an explicit error definition to minimize, are widely used for AUV control and can be precisely tuned to achieve the desired behavior [37, 39, 53, 62, 82, 83]. With the rise of deep reinforcement learning in robotics, there has also been recent work in using neural networks for AUV control [13, 67]. These controllers can execute complex behavior and are not limited to minimizing a concrete error signal. Although, as data-driven methods, they require substantial data to

learn an effective control policy.

In this thesis, we explore the problem of diver following with an AUV. Inspired by recent advancements in vision-based diver perception, we directly control the motion of our AUV using visual feedback from an onboard camera, a technique known as visual servoing [51]. Following this technique, we modularize our system, separating it into distinct vision and control modules. The vision module processes the incoming images and perceives the relative location of the diver in the field of view (FOV). We do not use fiducial markers, beacons, or other tools to aid tracking, maintaining a general problem formulation. Taking input from the vision module, the control module is responsible for taking actions to keep the diver within the FOV despite dynamic diver movements and external disturbances.

In an alternative approach, we could have bypassed the vision module and adopted an end-to-end framework where the controller would take actions directly from the incoming images. This approach would allow full flexibility to implicitly construct or learn a control policy without requiring feature extraction or a precise error signal definition. However, it would struggle to generalize since minor changes in diver appearance or modifications to the control system would necessitate replanning or retraining of the entire system. In our modular approach, we can modify the vision or control modules without significant changes to the other. This allows us to consider and evaluate different diver detection and control paradigms. It also provides a flexible formulation that enables us to extend our system to other visual ser-

voing tasks, beyond the scope of diver following. The main disadvantage of having separated vision and control modules is their explicit dependency on each other. Weaknesses in the one module will be propagated to the other.

To realize our autonomous diver following system in practice, we deploy both our vision and control modules onboard an AUV named Aqua [31, 38]. Aqua, shown in Figure 1.1, is a highly maneuverable hexapod robot with six fins mounted alongside its body. By strategically moving its fins, Aqua can quickly change its pitch, yaw, and roll angles as well as its linear velocity. Although highly maneuverable, Aqua exhibits complex dynamics due to its hexapod design with interdependent fins. It can be challenging to control in underwater environments compared to other AUVs that use propellers or rudders for motion.

For perception, Aqua is equipped with a front-facing monocular camera and an NVIDIA Jetson Xavier NX board to extract and process images from its surrounding environment. For control, Aqua includes a low-level control module that receives high-level user commands and interfaces with hardware elements to achieve the desired behavior. The high-level commands are unitless values within the range $[-1, 1]$ for pitch, roll, and yaw, and $[0, 1]$ for linear velocity. Working with the high-level commands and considering our problem of diver following, we send pitch rates to have Aqua swim up or down, yaw rates to swim left or right, and linear velocity rates to swim faster or slower. To avoid variations in the roll angle that are not required for our problem and could result in undesired behavior, such as flipping upside down,

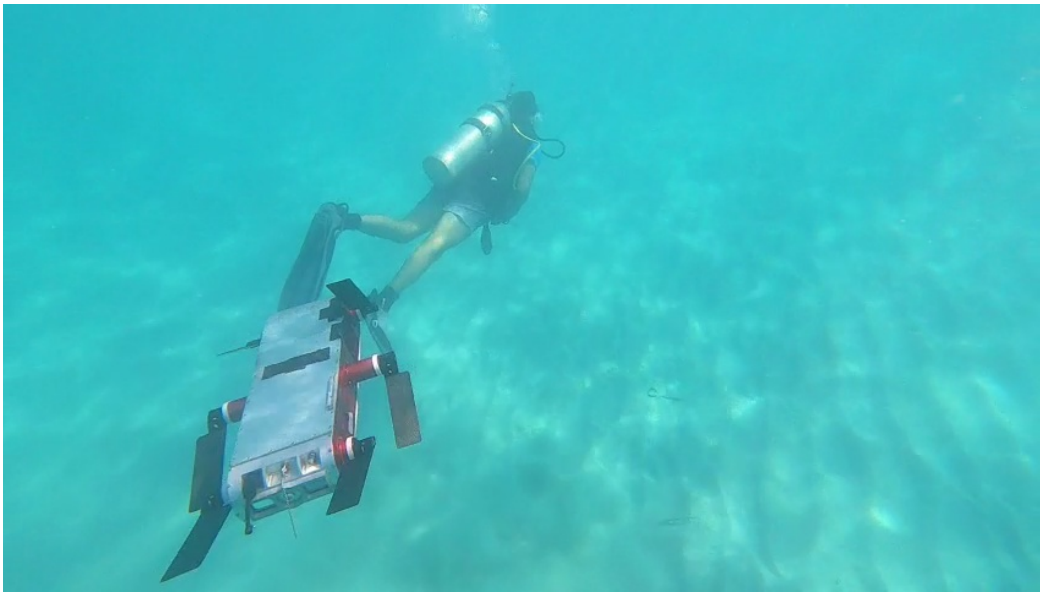


Figure 1.1: Aqua, a hexapod robot used in our autonomous scuba diver following experiments.

we integrate a simple yet effective roll stabilizer into our control module.

To conduct controlled and repeatable experiments, we use a commercial underwater simulator [52] based on the Unity game engine [90]. The simulator, shown in Figure 1.2, mirrors the hardware control stack used in practice and approximately models the complex dynamics of Aqua, including thrust generated from fin movement and hydrodynamic drag [40]. It models buoyancy, allowing us to construct realistic scenarios where Aqua operates as positively or negatively buoyant in its environment. Additionally, it models currents and wave motions, allowing us to apply external forces to Aqua during operation.

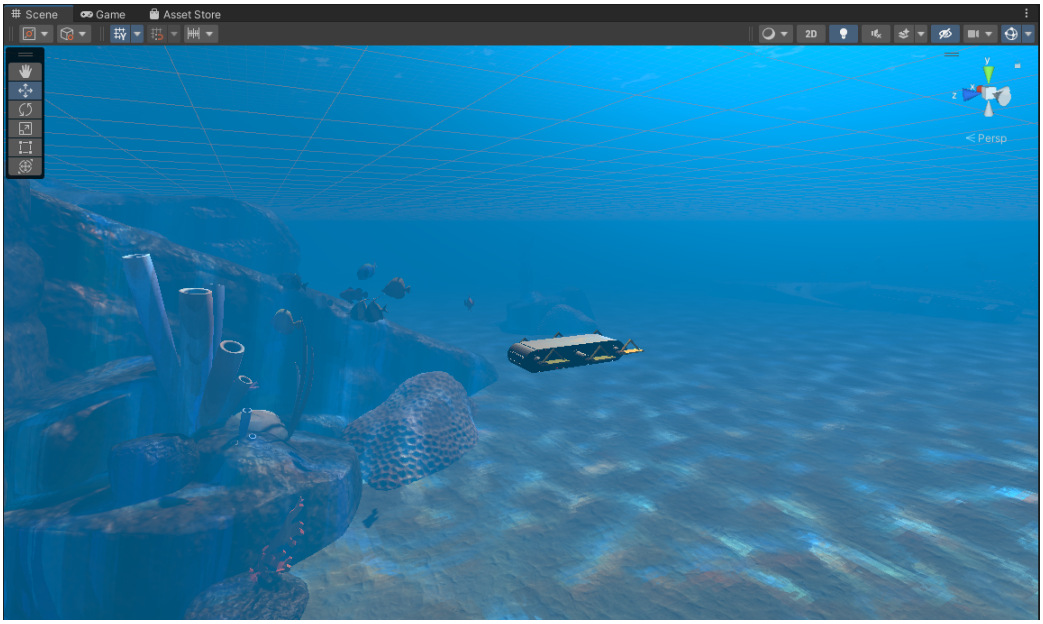


Figure 1.2: Our Unity-based simulation environment used for development and controlled experiments.

1.1 Outline

The remainder of this thesis is organized as follows. In Chapter 2, we introduce our vision module to robustly detect and track scuba divers. We explicitly address scenarios such as missing and false detections, which can be catastrophic in a robotic visual servoing system. In Chapter 3, we explore control paradigms, starting with traditional PID controllers, then extending to the potential for a learning-based controller within the context of diver following. Finally, we conclude this thesis in Chapter 4 with a comprehensive discussion on our findings and identify avenues for future research.

2

Vision module

Implementing a modular visual servoing strategy for diver following requires designing a robust, stable, and efficient vision module. Integrated directly into the control loop, this module must ensure that predicted diver locations are not only precise but also temporally stable across frames. Fluctuations in diver visibility and inconsistent predictions can severely impede the controller's ability to function in such unstable scenarios. In this chapter, we introduce our vision module, which combines a state-of-the-art deep learning-based model for diver detection with a lightweight filtering algorithm for tracking. To begin, we explore relevant work in object detection and tracking, presenting an overview of both traditional and data-driven methods tailored to underwater applications. Next, we discuss our detection module, where we analyze and benchmark the performance of seven object detection

algorithms and develop a dataset of simulated divers for use in simulation studies. Finally, we assess the temporal stability benefits of integrating a multi-object tracking algorithm alongside our detection module. We demonstrate that existing metrics for evaluating temporal stability are inadequate for real-time deployment scenarios and propose necessary adjustments.

2.1 Related work

In the computer vision community, the pursuit of robust object detection and tracking methods has been a cornerstone of research and development. Object detection, which includes both classification and localization, has seen a rich history of methodologies, ranging from traditional algorithmic techniques to deep learning and data-driven approaches commonly seen in modern day applications. The availability of large-scale, annotated datasets has propelled the advancement of deep learning-based object detection models, enabling them to learn rich representations that can transfer and generalize to a multitude of scenarios. Object tracking encompasses detection across multiple frames, focusing on methods to maintain object identities and estimate motion. In this section, we explore the general landscape of object detection and object tracking, with a particular focus on our application of scuba diver tracking.

2.1.1 Object detection

Object detection is a fundamental task in computer vision, involving both object classification and localization within an image. This task answers two main questions: *what* is the object and *where* is it located within an image. Classification answers the *what* question by identifying the object from a predefined list of classes. Localization answers the *where* question, typically by providing bounding box coordinates around the detected object. Object detection has been extensively studied [104], with methods dating back well before the prevalence of deep learning and data-driven approaches. These traditional methods typically split an input image into cells and perform feature extraction and classification within each cell. Viola and Jones [92] proposed a framework combining Haar-like feature descriptors with cascading classifiers for face detection. These cascading classifiers form a sequence of increasingly complex image classifiers. Image cells pass through each classifier, as shown in Figure 2.1, only continuing if positively classified. This enables high-speed face detection, as negative cells can be quickly discarded by the weak classifiers. Dalal and Triggs [21] combined Histogram of Oriented Gradients (HOG) feature descriptors with a linear support vector machine (SVM) for human detection. HOG feature descriptors capture the distribution of gradient orientations within image cells and can robustly describe an object's shape and boundaries. Lowe [66] introduced Scale Invariant Feature Transform (SIFT) features for object detection. SIFT features, by construc-

tion, are invariant to scale, translation, and rotation. Unlike dense HOG features, SIFT features are local and sparse and can be used as robust keypoints for object detection.

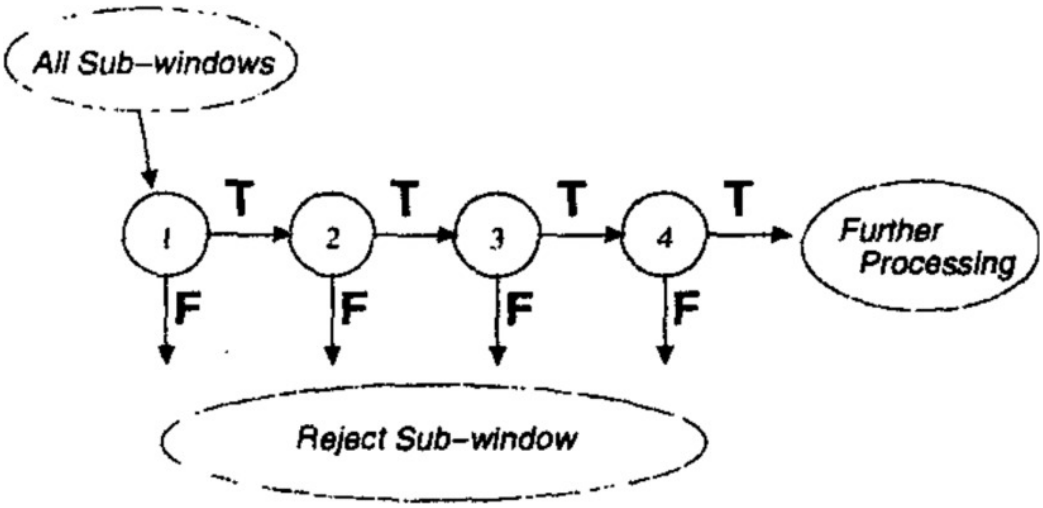


Figure 2.1: A diagram of cascading classifiers used by Viola and Jones [92] for high-speed face detection. Image cells or sub-windows pass through a series of classifiers, only continuing if positively classified. The initial classifiers can eliminate a large number of negative cells with minimal processing time. Reprinted from [92].

In the past decade, deep learning-based object detectors have gained enormous traction and are widely considered the standard for modern-day object detection [104]. These data-driven approaches employ neural networks for both classification and localization subtasks and have outperformed traditional methods on many benchmark datasets. The family of deep learning-based object detectors can be categorized into one-stage and two-stage detectors. One-stage detectors directly perform classification and localization from extracted features, achieving remarkable inference speeds suitable for

real-time deployment. Two-stage detectors incorporate an additional region proposal layer between feature extraction and classification and localization steps. This region proposal layer helps improve detection accuracy, at the cost of having slower inference speeds.

Convolutional Neural Networks (CNNs) are commonly employed as feature extractors in both one-stage and two-stage detectors. These networks learn complex patterns and structures within images through successive convolutional layers with learned filters, non-linear activation functions, and pooling layers to aggregate information across local regions. You Only Look Once (YOLO) detectors are a family of one-shot CNN-based architectures that have been widely adopted for real-time object detection tasks. YOLO was introduced in 2016 by Redmon et al. [74] and since then has had frequent version updates [88], the most recent being YOLOv9 [94]. Figure 2.2 shows a timeline of YOLO version updates from 2015 to 2023. The Single-Shot Multi-Box Detector (SSD) [65] is another popular one-shot object detection algorithm that is commonly combined with a MobileNet CNN backbone [50] for optimized real-time detection on embedded devices. In the two-stage realm, Region-Based Convolutional Neural Networks (R-CNN) [42] have been proposed, with iterations Fast R-CNN [41] and Faster R-CNN [76] introduced to improve inference speeds.

Aside from CNN-based architectures, there has been a recent surge in applying the transformer architecture [91], originally proposed for natural language processing, to object detection and other computer vision tasks.

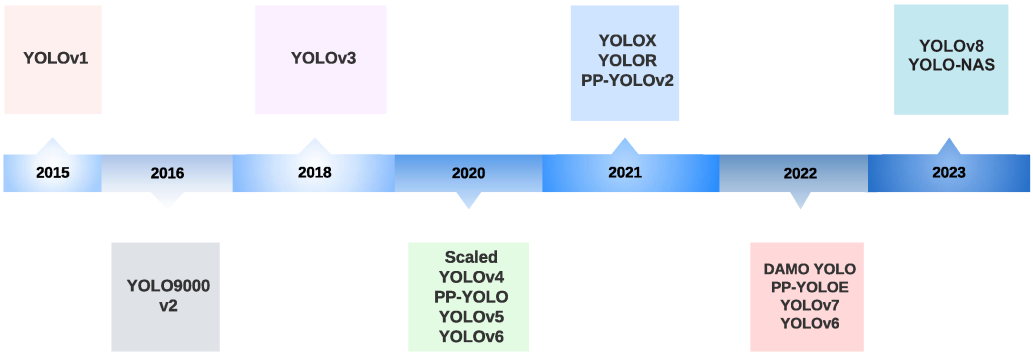


Figure 2.2: Evolution of the YOLO object detection algorithm. Reprinted from [88].

Rather than processing raw pixels directly, these architectures partition images into patches, treating each patch as a token. They utilize self-attention mechanisms, rather than pooling layers, to capture global context by considering relationships among all patches. The Vision Transformer (ViT) model [28], introduced in 2020, has opened the door for popular models such as the Detection Transformer (DETR) [12] and the Pooling-based Vision Transformer (PiT) [49], which both use the transformer architecture for object detection.

In the marine robotics community, both traditional and deep learning-based methods have been proposed for scuba diver detection. Sattar and Dudek [80] introduced an algorithm for diver detection by identifying periodic motion associated with kicking. This method divides an image into cells and analyzes each cell in the frequency domain to detect flipper oscillations. Several improvements have been proposed, including the use of a flipper color prior to prune the search space [54] and extending to detect motion

in multiple directions [81]. Chavez et al. [14] proposed an algorithm for diver detection using a modified random forest classifier on image cells. This algorithm is optimized to minimize memory and performance issues when scaling the number of features used for classification.

Exploring the potential of deep learning-based methods, Xia and Sattar [98] used Faster R-CNN to detect divers and subsequently extracted features from each detection for diver identification. Although this method achieved impressive results on collected datasets, the computational load of a two-stage detector with additional feature extraction remains significant, and its feasibility in real-time applications was not fully explored. In contrast, Islam et al. [53] designed a custom CNN-based architecture optimized for single diver detection, achieving inference speeds suitable for real-time deployment. Fulton et al. [37] proposed an algorithm to approach scuba divers with an autonomous underwater vehicle (AUV), utilizing YOLOv4 [8] for the detection component. Expanding the scope to the broader task of underwater object detection, Shkurti et al. [83] developed a vision-based framework for an underwater multi-robot convoy, as shown in Figure 2.3. This method leveraged a modified version of YOLOv2 [75] to detect and track robots in the image plane.

2.1.2 Datasets

A critical factor in the performance of deep learning-based object detectors is the quantity and quality of the data they are trained on. Insufficient and

Figure 2.3: An underwater multi-robot convoy using a modified version of YOLOv2 for robot detection. Reprinted from [83].

repetitive data can lead to poor generalization and overfitting of the models to the training set. Fortunately, there continues to be a considerable growth in the amount of available image data, allowing for the coalition of large, diverse datasets for data-driven methods to leverage. The PASCAL Visual Object Classes (VOC) challenges, particularly the 2007 [33] and 2012 [34] challenges introduced two benchmark datasets for object detection. Both datasets contain annotations for 20 objects commonly seen in everyday life. The VOC2007 dataset includes close to 10,000 images, while the VOC2012 dataset includes over 17,000 images.

In 2009, ImageNet [26], a large-scale visual database, was introduced

for image classification and object detection tasks. It currently contains over 1 million images with bounding box annotations. Subsets of ImageNet have been used in the ImageNet Large Scale Visual Recognition Challenges (ILSVRC) [79], serving as popular benchmarks in image classification and object detection. The Microsoft Common Objects in Context (MS COCO) [64] dataset, introduced in 2014, currently contains over 200,000 labeled images. Although smaller in terms of the number of images compared to ImageNet, it includes a notable 1.5 million object instances across 80 categories within those images.

The compilation of these datasets has revolutionized transfer learning and fine-tuning approaches in deep learning-based object detection. Training on diverse, large-scale datasets such as ImageNet or MS COCO allows models to extract rich, generalizable information from images. These learned representations can be used in transfer learning schemes, serving as a feature extractor for training custom detection modules. They can also serve as a strong starting point that can be fine-tuned for target applications.

Utilizing these learned representations can significantly reduce training time and computational requirements as less data is required in the target domain to achieve strong performance. Additionally, transfer learning and fine-tuning approaches have been shown to outperform training from scratch, showing enhanced performance and better generalization to out-of-distribution samples [15, 86]. Consequently, they have become practical and preferred approaches for training deep learning-based object detectors.

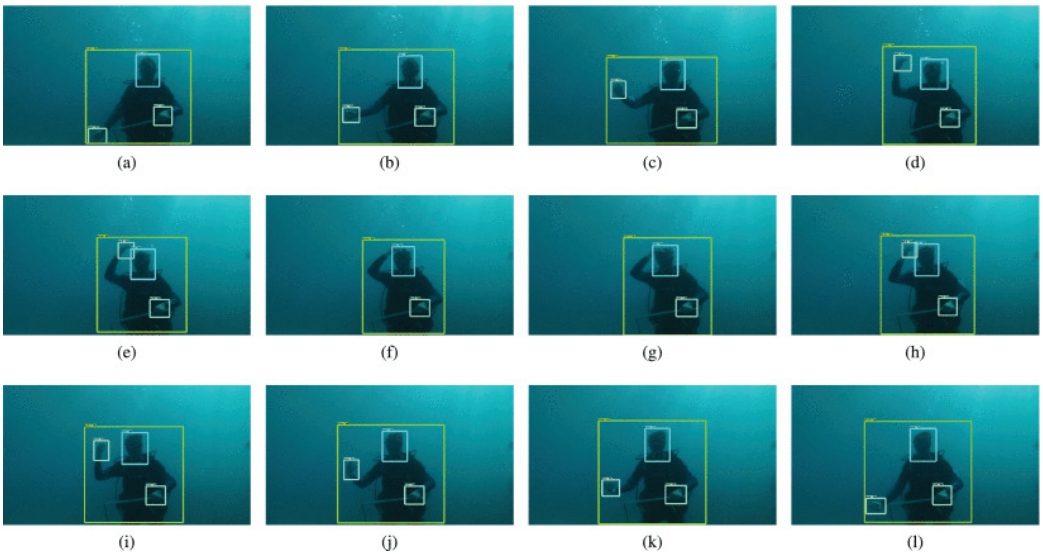


Figure 2.4: Sample images from SCUBAnet, a dataset of scuba divers tailored for hand signal and body language recognition. Reprinted from [19].

In the context of underwater diver detection, recent efforts have focused on compiling large datasets of scuba divers in both pool and ocean environments. Perhaps the most significant contribution in this domain is the Video Diver Detection (VDD-C) dataset [60], containing over 100,000 labeled images of divers captured in oceanic and indoor pool settings. The dataset is extracted from video sequences, enabling the evaluation of second-order temporal stability metrics alongside accuracy. Codd-Downey and Jenkin [19] introduced SCUBAnet, a dataset featuring over 1,000 images of divers in freshwater and saltwater environments. This dataset provides annotations for each diver's body, head, and hands, as shown in Figure 2.4. It is tailored to advance diver-robot communication methods through hand signal and body language recognition [20].

Further contributing to the field of diver-robot interaction, the Cognitive Autonomous Diving Buddy (CADDY) project has produced two notable datasets [43]. The first, the CADDY Underwater Diver Pose dataset, contains 12,000 images of divers annotated with heading measurements. The second, the CADDY Underwater Gestures dataset, includes 10,000 images of divers displaying various hand signals.

2.1.3 Object tracking

Unlike object detection, which aims to classify and localize objects within a single frame, object tracking involves the sequential identification and motion estimation of an object across multiple frames. Object tracking algorithms complement object detection modules by removing false detections, estimating missed detections, and smoothing detections between consecutive frames. These algorithms broadly fall into two categories: single-object trackers and multi-object trackers.

Single-object trackers focus on tracking a single target throughout a video sequence, maintaining its identity and estimating its motion over time. They address challenges such as changes in appearance, scale variations, and occlusions. The Kernelized Correlation Filter (KCF) [48] is a well-known single-object tracker, utilizing correlation filters to learn a mapping between input image features and target locations. Notably, it uses the kernel trick to handle non-linearities in data, and Fourier transforms for efficient computation in the frequency domain. Other popular single-object tracking methods based

on correlation filters include the Discriminative Scale Space Tracker (DSST) [22] and the Minimum Output Sum of Squared Error (MOSSE) tracker [9].

In recent years, deep learning-based single-object trackers have gained attention due to their ability to learn robust feature representations and generalize well to changes in object appearance or background environments. These algorithms often employ Siamese networks to learn similarity metrics between pairs of inputs [6, 45, 87], facilitating robust matching between the target object and candidate image patches during tracking.

In contrast, multi-object trackers simultaneously track multiple objects within a video sequence. They face similar challenges to single-object trackers, but must also address the complex data association problem of matching incoming detections with the correct objects and maintaining consistent identities over time. Multi-object trackers can be categorized into batch-based and online algorithms. Batch-based algorithms [17, 27, 100] process video sequences in batches of frames and are suited for offline tracking, where both future and previous frames are available. Online algorithms assume the tracker only has access to detections in the current and previous frames, making them suitable for real-time object tracking. Simple Online and Realtime Tracking (SORT) [7] is a widely used multi-object tracker that uses Intersection over Union (IoU) similarity for data association, and a Kalman filter for motion estimation. Due to its simple data association metric, SORT can process incoming detections with minimal computational overhead, making it a desirable choice for real-time tracking.

Recent iterations of SORT such as DeepSORT [97], StrongSORT [29], and BoT-SORT [3], integrate object appearance in addition to location similarity for improved data association. These iterations incorporate neural networks for feature extraction, resulting in improved tracking performance and fewer identity switches between closely located objects. However, they come with the additional computational cost of deep learning-based feature extraction.

Scuba divers swimming in the open ocean are highly dynamic objects with erratic movements. Visually tracking these divers requires concrete steps to ensure temporal stability and accurate motion estimation. Sattar and Dudek [81] utilized an unscented Kalman filter (UKF) to estimate and track diver motion once detected in the image plane. Langis and Sattar [61] investigated the application of DeepSORT for diver tracking and re-identification, analyzing its performance on offline video sequences. Agarwal et al. [2] used Long Short-Term Memory (LSTM) networks to predict and stabilize diver trajectories from the VDD-C dataset. Additionally, in a multi-robot convoy application, Shkurti et al. [83] designed a custom LSTM architecture to enhance the temporal stability of incoming detections and reduce tracking failure rates.

2.2 Detection module

Following recent advancements in deep learning-based object detection, we explore the potential of seven deep architectures for diver detection. Previous work in underwater object and diver detection has predominantly leaned

towards YOLO models, with Langis et al. [60] recommending them over alternative one-shot detectors for real-time inference. At the time of this study, YOLOv4 [8] was the most recent YOLO version used in prior work for scuba diver detection. It was used in a system by Fulton et al. [37] designed to autonomously approach scuba divers. Building upon this, we assess the potential of newer YOLO versions, namely YOLOv5 [55] and YOLOv7 [93], which was the latest version available at the time of this study.

In response to the emerging interest in using vision transformers for object detection, we are intrigued by how these models compare with CNN-based architectures. Thus, we also explore the potential of the Detection Transformer (DETR) model. The DETR model comprises a CNN backbone for feature extraction, an encoder-decoder transformer, and four feed-forward networks, which are 3-layer perceptrons responsible for generating the final predictions. In our investigation, we consider the base DETR model, as well as the DETR-DC5 variant, which incorporates dilation in the final stage of the backbone to increase resolution for detecting small objects. For both models, we consider the use of both ResNet50 (R50) and ResNet101 (R101) as the CNN backbone [47].

2.2.1 Evaluation metrics

Keeping in mind the real-time requirements of our system, our evaluation encompasses both the accuracy and inference speed of each object detection model. For accuracy assessment, we employ the widely used mean average

precision (mAP) metric. This metric calculates the average precision (AP) for each object class and averages them together to yield the final result. In our application we detect only a single object class, namely a scuba diver, thus mAP simplifies to AP. In terms of inference speed, we measure the frames processed per second (FPS) during inference.

Deriving the AP metric requires a clear understanding of both detection confidence and Intersection over Union (IoU). Detection confidence is a measure of prediction certainty, represented as a probability attached to each bounding box prediction. Setting a threshold on detection confidence allows low-confidence detections to be filtered out. IoU quantifies the alignment between predicted and ground truth bounding boxes. IoU values range from 0 to 1, where 0 signifies no overlap between a prediction and the ground truth, and 1 signifies perfect alignment. IoU thresholds are used to derive true positive (TP), false positive (FP), and false negative (FN) rates. A prediction is categorized as a true positive if its overlap with the ground truth exceeds the IoU threshold; otherwise, it is considered a false positive. If no prediction surpasses the IoU threshold for a ground truth bounding box, the false negative rate increases.

By setting IoU and detection confidence thresholds, we can derive precision and recall values. Precision, defined in Equation 2.1, measures how accurately the model detects target objects. Increasing the confidence threshold results in higher precision as the model only outputs its most confident and accurate predictions. Recall, defined in Equation 2.2, measures the abil-

ity of the model to find all target objects. Increasing the confidence threshold results in lower recall as many low confidence detections get filtered out.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

Taking an IoU threshold as input, we calculate AP by first computing precision and recall values across multiple confidence thresholds. An ideal object detection model achieves both high precision and high recall across a range of confidence thresholds. To gauge this balance, we plot a precision-recall (PR) curve, taking the area under the curve as AP_t, the average precision at the IoU threshold *t*. A larger AP signifies strength in both precision and recall.

2.2.2 Datasets

To train our selected models for diver detection, we adopt a supervised learning approach where our models learn to map input images to bounding box coordinates of detected divers using annotated data. Following this approach, we require a diverse dataset comprising images of scuba divers along with their respective locations in the image plane. The VDD-C dataset [60] is a natural candidate, offering a substantial collection of 104,838 labeled im-

ages of scuba divers. This dataset not only boasts a large quantity of divers but also exhibits considerable variation, featuring divers at diverse locations, depths, and orientations.

The VDD-C dataset includes rich scenarios, containing images with a single diver, multiple divers, as well as images with no divers. A sample image containing three divers is presented in Figure 2.5. The dataset encompasses both oceanic and indoor pool settings, reflecting real-world conditions and enabling generalization across different environments. As an added benefit, the oceanic segment was gathered off the west coast of Barbados, roughly in the same region where we conduct our field experiments.

Figure 2.5: A sample annotated image from the VDD-C dataset.

For consistent evaluation and comparison of diver detection models, the VDD-C dataset comes pre-split into training, testing, and validation sets.

Table 2.1 provides a detailed breakdown of these splits. Each set contains images extracted from video sequences, simulating the practical scenario of images incoming from a video stream. This setup enables the evaluation of not only accuracy but also temporal stability, which is a crucial consideration for real-world deployment.

Figure 2.6: A sample annotated image from the SDD dataset.

To conduct insightful simulation studies, we adopt a mirrored pipeline for diver detection in simulation. In particular, we utilize an object detection module to extract the location of the diver, rather than projecting the ground truth location of the diver, which is available in the simulation, into the image plane. In our setup, we introduce a basic swimming character, shown

in Figure 2.6, positioned within our simulated underwater environment. As the optics of our simulated diver differ significantly from real-world divers, we collected and labeled a custom dataset, which we refer to as the Simulated Diver Detection (SDD) dataset, for use in our simulation experiments.

	Training	**Validation**	**Test**	**Total**
VDD-C	71,591	14,475	18,772	104,838
SDD	1,331	110	162	1,603

Table 2.1: A breakdown of the training, testing, and validation splits in the VDD-C and SDD datasets.

We collected the SDD dataset by setting random velocities for the simulated diver and manually controlling Aqua to capture photos. As depicted in Table 2.1, the SDD dataset contains 1,603 images, each annotated with a bounding box around the simulated diver. To explore the diversity of our collected dataset, we analyze the distribution of diver locations and sizes. Figure 2.7 illustrates that the diver is located across various locations in the image plane, with a bias towards the center. This bias likely results from our efforts to keep the diver stable in the image during data collection. Figure 2.8 reveals that the diver area primarily occupies less than 10% of the image plane, indicating a shortage of images with the diver in close proximity in our collected dataset.

To further enhance the diversity of the SDD dataset, we perform data augmentation on the training set. Data augmentation involves generating modified samples from existing samples to artificially increase training data.

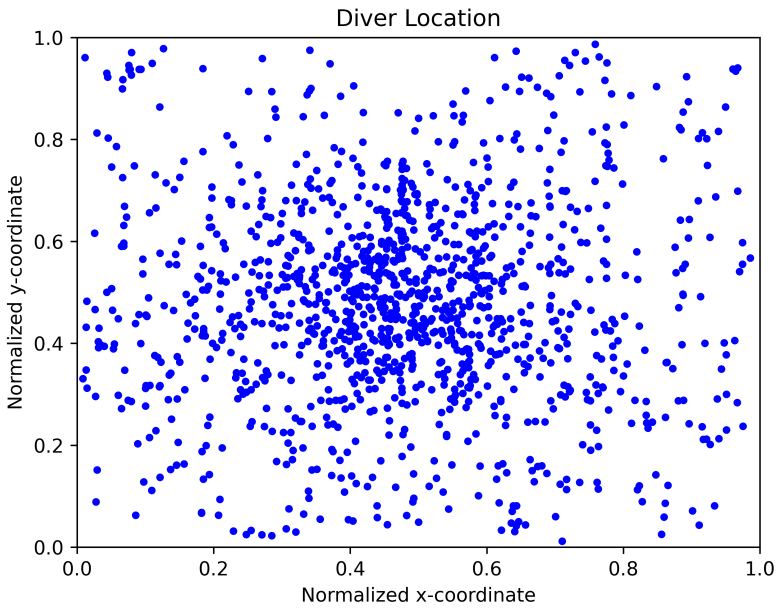


Figure 2.7: Diver location distribution in the SDD dataset, showcasing diversity across the image plane.

Common augmentation techniques in image processing include horizontal and vertical flipping, random rotations, brightness and color adjustments, and blurring. Not only does data augmentation expand the dataset size, but it facilitates learning generalizable information and helps prevent overfitting by exposing the model to a diverse range of samples. For the SDD dataset, we perform random flips, rotations, crops, blurs, as well as brightness, saturation, and hue adjustments. These data augmentation techniques result in a training set of 3,993 images for the SDD dataset, approximately three times the original size.

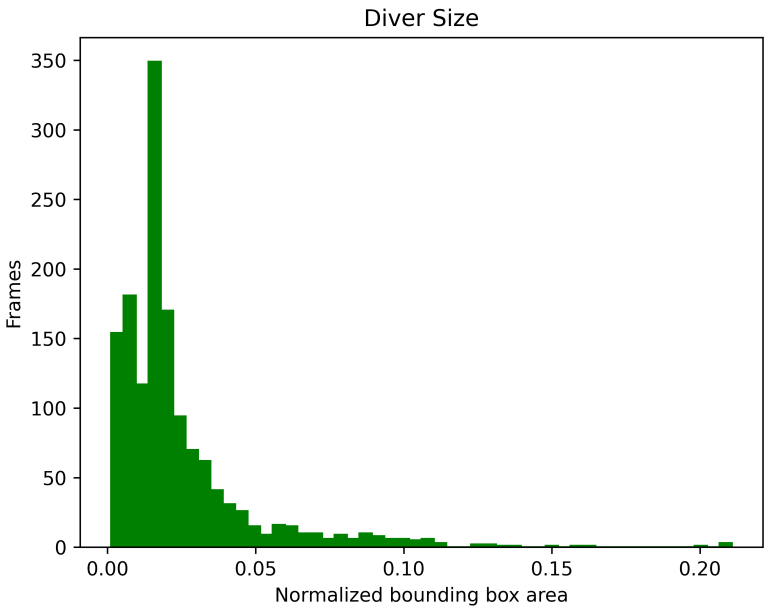


Figure 2.8: Diver size distribution in the SDD dataset, revealing a shortage of images with the simulated diver in close proximity.

2.2.3 Training

Deep learning-based object detectors typically release pretrained models, trained on extensive datasets such as ImageNet or MS COCO. In particular, our selected YOLO and DETR models are available pretrained on the MS COCO dataset. Leveraging these pretrained models, and their learned representations, can significantly reduce data and computational requirements while improving performance and generalization compared to training from scratch. Thus, to utilize these pretrained models, we consider fine-tuning or transfer learning techniques to train our selected models for diver detection.

In transfer learning, the pretrained model acts as a feature extractor, with the foundational layers frozen and only the detection head or the final layers being trained. This approach proves beneficial in cases where data in the target domain is limited, as only a subset of the model's layers need to be trained. However, given the ample data available in our scenario through the VDD-C dataset, we opt for a fine-tuning approach, allowing all layers of the model to be updated during training.

Our training procedure adheres to standard supervised learning practices. We fine-tune each selected model on the training set, utilizing the validation set for hyperparameter tuning and early stopping to prevent overfitting. Similar to Shkurti et al. [83], we resize input images to 416 x 416 pixels to reduce the computational load during image processing. Evaluation is performed on a held-out test set, ensuring unseen data is used for assessment. Each selected model undergoes fine-tuning on the VDD-C dataset, with performance evaluated based on both inference speed and accuracy.

An important note is that we exclusively use the VDD-C dataset, not the SDD dataset, for training and comparison of our selected diver detection models. After analyzing the results and selecting an architecture to deploy in practice, we then fine-tune the chosen architecture on the SDD dataset. This decision ensures the simulated pipeline mirrors the real-world pipeline, facilitating seamless integration of code and unbiased simulation studies.

2.2.4 Comparison study

The results of our comparison study are presented in Table 2.2. In terms of accuracy, we compute AP at IoU thresholds of 0.5 ($AP_{0.5}$), 0.75 ($AP_{0.75}$), and an average of 10 thresholds evenly spaced between 0.5 and 0.95 ($AP_{0.5:0.95}$). Higher IoU thresholds present a stricter criterion for classifying true positives; thus, AP decreases as the IoU threshold increases. $AP_{0.5:0.95}$ offers insights into detection performance across a spectrum of IoU thresholds. In terms of inference speed, we measure the FPS during inference using a single NVIDIA GeForce GTX 1080 Ti GPU. Note that this GPU is significantly more powerful than the NVIDIA Jetson Xavier NX board equipped on Aqua. Thus, although all models in Table 2.2 showcase strong inference speeds, it is important to consider their relative performance as we anticipate a noticeable drop in performance when deployed on the Jetson Xavier NX.

Our first observation from this study is the strength of transformer-based architectures. The DETR models, in particular the DETR-DC5-R101 model, provides competing performance with YOLO. However, a notable trade-off emerges as the DETR models struggle to maintain inference speeds comparable to YOLOv5 and YOLOv7. The DETR-DC5 models, which include dilation in the final stage of the CNN backbone, exhibit sightly slower inference speeds compared to their respective counterparts. Additionally, models utilizing a ResNet101 backbone exhibit longer inference times than those with a ResNet50 backbone, aligning with the expected computational overhead of

a larger backbone.

Within the YOLO family of models, noteworthy improvements are observed, with YOLOv4 surpassing the previously reported AP values in the VDD-C benchmark [60]. Subsequent iterations demonstrate further advancements, as YOLOv5 outperforms YOLOv4, and YOLOv7 outperforms YOLOv5 across all IoU thresholds.

Among all considered detection models, YOLOv7 achieves the highest AP across all IoU thresholds while maintaining the second highest FPS, surpassed only by its predecessor, YOLOv5. Factoring in both accuracy and inference speed, we opt to integrate YOLOv7, trained on the VDD-C dataset, as our diver detection module.

	$\mathbf{AP_{0.5}}$	$\mathbf{AP_{0.75}}$	$\mathbf{AP_{0.5:0.95}}$	**FPS**
DETR-R50	0.904	0.507	0.508	40
DETR-DC5-R50	0.887	0.476	0.487	38
DETR-R101	0.903	0.487	0.496	27
DETR-DC5-R101	0.911	0.499	0.505	26
YOLOv4	0.898	0.431	0.468	29
YOLOv5	0.894	0.578	0.538	**107**
YOLOv7	**0.931**	**0.615**	**0.566**	70

Table 2.2: Evaluation of seven deep learning-based object detection models on the VDD-C dataset.

After selecting YOLOv7 as our diver detection module for deployment in practice, we fine-tune it on our augmented SDD dataset to ensure a mirrored pipeline in simulation. The results are summarized in Table 2.3. Notably, YOLOv7 demonstrates near perfect performance across the chosen IoU

thresholds, and achieves inference speeds similar to those observed during the processing of real-world images. These findings suggest that our diver detection task in simulation is easier than in practice. Thus, we should anticipate a drop in performance when deploying our system and consider techniques to handle cases of missing and false detections.

	$\mathbf{AP_{0.5}}$	$\mathbf{AP_{0.75}}$	$\mathbf{AP_{0.5:0.95}}$	**FPS**
YOLOv7	0.995	0.985	0.890	75

Table 2.3: Evaluation of YOLOv7 on the SDD dataset.

2.3 Tracking module

To improve the temporal stability of our vision module, and explicitly address missing and false detections, we integrate the Simple Online and Realtime Tracking (SORT) algorithm alongside YOLOv7. SORT is a lightweight multi-object tracking algorithm, proficient at tracking objects across frames with minimal computational burden. The algorithm follows a two-step process of data association and filtering. During data association, detections in the current frame are matched with tracked objects. SORT accomplishes this by first computing an assignment matrix of pairwise IoU similarities between the predicted locations of each tracked object, and the incoming detections. Then, SORT efficiently obtains the optimal assignment using the Hungarian algorithm [58]. In the subsequent filtering step, SORT uses a Kalman filter with a constant velocity model to update tracked objects with the matched detections. This enables detections in the current frame to be updated based

on corresponding detections from previous frames.

We chose SORT instead of single-object tracking algorithms to easily scale to applications that involve tracking multiple divers. Given the composition of the VDD-C dataset, which includes frames with multiple divers, selecting a multi-object tracking algorithm with an explicit data association mechanism seemed a natural choice. Among multi-object tracking algorithms, we did not consider batch-based trackers because of their lack of real-time tracking capabilities. Considering online trackers, SORT emerged as the preferred choice over its successors, DeepSORT and BoT-SORT, which incorporate object appearance during data association. This decision was primarily influenced by the fact that SORT uses only IoU similarity for data association, rather than a deep association metric, enabling tracking with minimal computational overhead. Considering our practical deployment goals, we did not want to introduce any additional computational load unless necessary. Furthermore, our current focus does not lie in diver identification or mitigating identity switches between closely located divers; thus, directly scaling to a deep association metric seemed unnecessary.

2.3.1 Effects on detection accuracy

Two hyperparameters of particular importance in SORT are *min_hits* and *max_age*. The *min_hits* hyperparameter filters out false detections by requiring an object to be detected in *min_hits* consecutive frames before initializing a tracker for the object. The *max_age* parameter helps to fill in missed detec-

tions by outputting the tracked object's predicted location for a maximum of *max_age* consecutive frames without detection.

Both *min_hits* and *max_age* hyperparameters are promising tools for improving the temporal stability of our vision module. However, it is important to first explore their effect on detection accuracy. In our system, we combine vision and control, meaning that the output of our vision module can directly impact the performance of our controller. Thus, we need to ensure a balance of both accuracy and temporal stability. To evaluate the effect of these hyperparameters on detection accuracy, we perform an ablation study over 25 hyperparameter combinations, computing the average $AP_{0.5:0.95}$ across each of the eight video sequences included in the VDD-C test set. The results are presented in Table 2.4

		min_hits				
		1	5	10	20	30
max_age	1	**0.585**	0.571	0.549	0.500	0.458
	5	0.577	0.572	0.565	0.542	0.517
	10	0.576	0.572	0.565	0.544	0.520
	20	0.574	0.572	0.566	0.549	0.531
	30	0.572	0.571	0.565	0.552	0.536

Table 2.4: An ablation study over two important hyperparameters in SORT. The metric presented is $AP_{0.5:0.95}$ averaged over eight video sequences from the VDD-C dataset.

We observe for all hyperparameter combinations, detection accuracy is not significantly reduced by having SORT smooth detections between frames. In fact, for some combinations, AP is higher than the standalone YOLOv7

model because SORT can incorporate information from previous frames to yield better estimates and fill in missed detections. For a given value of *max_age*, we observe that increasing *min_hits* reduces AP. This is not surprising because SORT filters out objects with less than *min_hits* consecutive detections. Thus, objects entering the field of view (FOV) take longer to be recognized. For a given value of *min_hits*, we observe that increasing *max_age* results in two opposing behaviors. The first behavior occurs when *min_hits* is 1, where increasing *max_age* leads to a decrease in AP. This is due to SORT preserving any false positive in a single frame for *max_age* frames. The second behavior occurs when *min_hits* is greater than 1, where increasing *max_age* generally leads to an increase in AP, especially for higher values of *min_hits*. This is due to SORT correctly filling in missed detections on tracked objects, while requiring a minimum consecutive detection count to remove false positives.

From the results of this ablation study, we present two recommendations for tuning *min_hits* and *max_age* in practice. First, *min_hits* should be initialized to a low value to have a minimal impact on accuracy. If many false detections occur, *min_hits* can be gradually increased to filter them out. Second, *max_age* should be set based on the strength of the object detector in the environment. If objects are often missed and recaptured, *max_age* should be set to the average number of missed frames between detections. However, it is important to note that the larger *max_age* is, the longer SORT will output a tracked object's predicted location without a detection, potentially

providing a false track for the controller to follow. For reference, in our open ocean experiments, we used a *min_hits* of 4 and a *max_age* of 7.

2.3.2 Effects on temporal stability

Our primary motivation behind integrating SORT is to improve the temporal stability of our detection module. However, commonly used metrics such as AP, used to evaluate object detection modules, are not sufficient to assess temporal stability since they only consider detections within a single frame. To address this, Langis et al. [60] and Zhang and Wang [102] proposed three metrics to evaluate the temporal stability of object detectors: fragmentation error, scale and aspect ratio error, and translation error. In the following sections, we define each of these metrics and compute them for the eight video sequences in the VDD-C test set. A note before continuing, we use the term *tracklet* to refer to a sequential list of detections of a particular diver across a video sequence. An example of two tracklets is shown in Figure 2.9.

Fragmentation error

Fragmentation error concerns the temporal consistency of detections. If a diver is frequently detected in one frame but missed in the next, the fragmentation error increases. Conversely, if a diver is consistently detected (or consistently ignored) across frames, the fragmentation error decreases. We define fragmentation error in Equation 2.3. For each ground truth tracklet k in a video sequence with N tracklets, the fragmentation error is the average number of fragments per tracklet, normalized by the tracklet length l_k. A

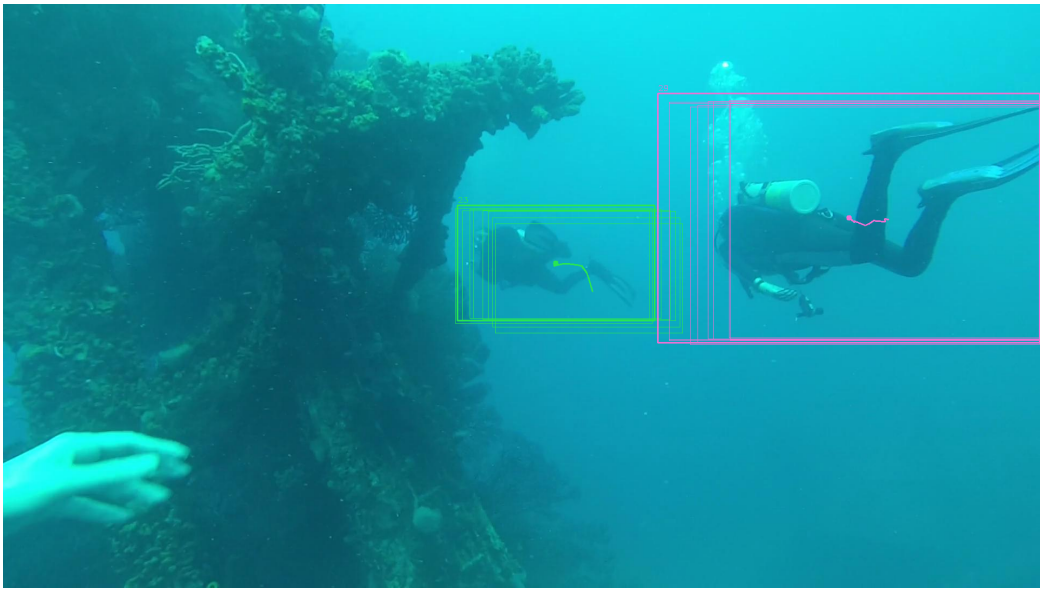


Figure 2.9: An illustration of two tracklets from a video sequence in the VDD-C dataset. A tracklet is a sequential list of detections of a particular diver across a video sequence. Each tracklet in the figure is shown in a different color, and we limit the size of the tracklets to 10 frames for clarity.

fragment f_k denotes the number of times a tracklet's status changes from detected to undetected, or vice versa.

$$\frac{1}{N}\sum_{k=1}^{N}\frac{f_k}{l_k - 1} \tag{2.3}$$

Scale and aspect ratio error

Scale and aspect ratio error concerns the stability of both scale and aspect ratio within a tracklet. If a diver is detected with inconsistent errors relative to the ground truth, the resulting scale and aspect ratio error will be high.

Conversely, the scale and aspect ratio error will be minimal if a diver is detected with consistent errors across the video sequence.

This metric, outlined in Equation 2.6, is computed for each video sequence by averaging the combined scale error and aspect ratio error of each tracklet. The scale error, $e_s(k)$, computes the standard deviation of the ratio of detected areas to ground truth areas for each detection d within tracklet k. The aspect ratio error, $e_r(k)$, similarly calculates the standard deviation of the ratio between the detected aspect ratio and the ground truth aspect ratio. In both Equation 2.4 and Equation 2.5, w_d and h_d represent the width and height of a detected bounding box, respectively, while w_g and h_g represent the width and height of the ground truth bounding box, respectively.

$$e_s(k) = \sigma\left(\sqrt{\frac{w_d h_d}{w_g h_g}}\right), \forall d \in k \tag{2.4}$$

$$e_r(k) = \sigma\left(\frac{w_d}{h_d} / \frac{w_g}{h_g}\right), \forall d \in k \tag{2.5}$$

$$\frac{1}{N}\sum_{k=1}^{N}(e_s(k) + e_r(k)) \tag{2.6}$$

Translation error

Translation error measures the stability of center position errors within a tracklet. In cases where a diver is detected with inconsistent center position errors compared to the ground truth, the resulting translation error will be high. If divers exhibit consistent center position errors across frames, the resulting translation error will be minimal. To compute the translation error, we average the combined center position errors along both the x and y dimensions, denoted as $e_x(k)$ and $e_y(k)$, respectively. We compute these center position errors using the standard deviation of the differences between the detected centers and the ground truth centers, in the dimension of interest. Equation 2.7 outlines our calculation of $e_x(k)$, and Equation 2.8 outlines our calculation of $e_y(k)$. In both equations, the subscript d refers to a detected bounding box, while the subscript g refers to a ground truth bounding box. The overall translation error is presented in Equation 2.9.

$$e_x(k) = \sigma\left(x_d - x_g\right), \forall d \in k \tag{2.7}$$

$$e_y(k) = \sigma\left(y_d - y_g\right), \forall d \in k \tag{2.8}$$

$$\frac{1}{N}\sum_{k=1}^{N}(e_x(k)+e_y(k)) \tag{2.9}$$

Results

Computing the above metrics requires access to ground truth tracklets and a method to associate detections with these tracklets in each frame. The VDD-C dataset lacks ground truth tracklets in its annotation set; therefore, we computationally generate them using a method proposed by Chen et al. [16], using IoU as the similarity metric. To associate output detections from our vision module to ground truth tracklets, we also employ IoU similarity. For each ground truth tracklet, in each frame, we pair the diver with the detection having the highest IoU similarity exceeding a threshold τ. If no detection is matched, this impacts the fragmentation error but not the scale, aspect ratio, and translation errors, as they only consider *matched* detections within a tracklet. In our computations, we set τ to 0.25.

We evaluate the temporal stability of our proposed YOLOv7+SORT model against the standalone YOLOv7 model. The evaluation is conducted across each video in the VDD-C test set, using the metrics defined above. For YOLOv7+SORT, we average the results over the 25 combinations of *min_hits* and *max_age* hyperparameters presented in Table 2.4, reporting both the mean and standard deviation.

The fragmentation errors for each video sequence are displayed in Figure 2.10. Across all sequences, we observe a substantial reduction in errors

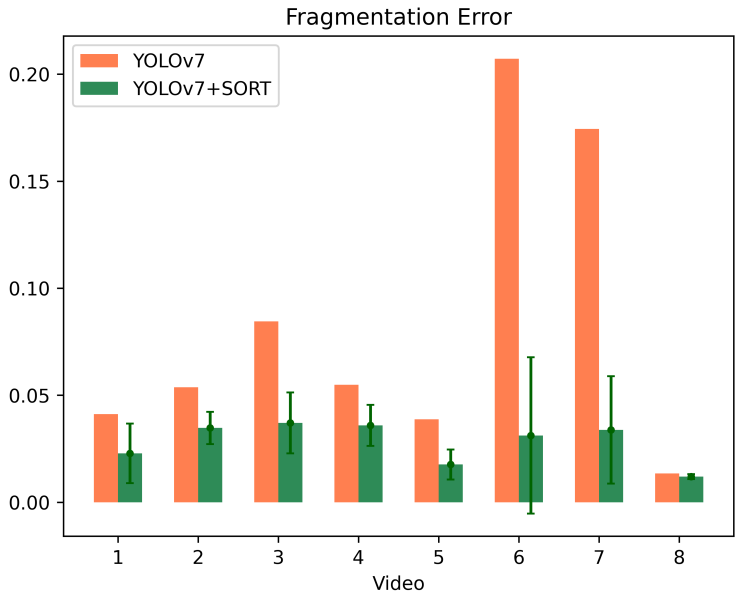
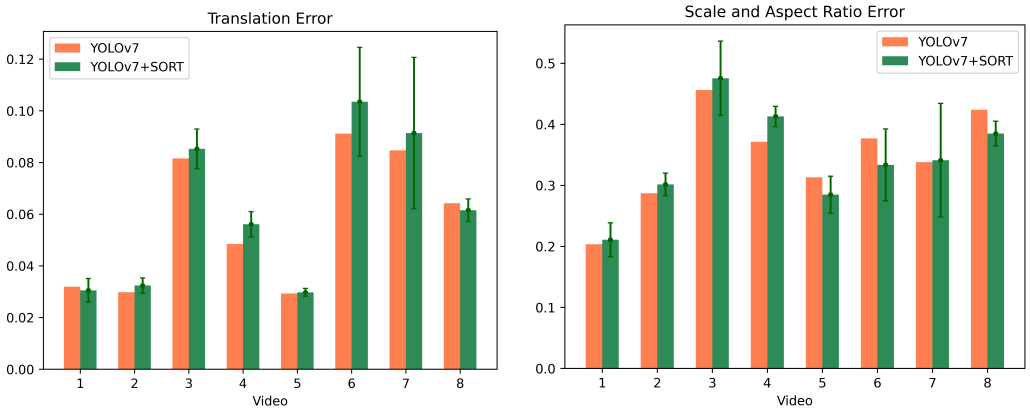


Figure 2.10: The fragmentation errors for each video sequence in the VDD-C test set, revealing a significant reduction when SORT is combined with YOLOv7.

when YOLOv7 is combined with SORT, with sequences 6 and 7 showing the most pronounced reductions. Upon closer examination of these two sequences, we observe particularly challenging conditions, including a blurred camera lens, severe camera instability, and frequent obstructions to the FOV from bubbles. In these scenarios, SORT's ability to incorporate temporal information and compensate for missed detections with predictions from the Kalman filter is key to maintaining consistent detections.

When analyzing the translation and scale and aspect ratio errors presented in Figure 2.11, the combination of YOLOv7 with SORT appears to *decrease* temporal stability with respect to these metrics. Although initially



(a) (b)

Figure 2.11: The translation errors (a) and scale and aspect ratio errors (b) for each video sequence in the VDD-C test set. Integrating SORT with YOLOv7 results in slightly larger errors when these metrics are computed with respect to the ground truth.

surprising, these results in fact align with the expected behavior of integrating SORT with YOLOv7. The reasoning behind this is that SORT performs temporal smoothing of detections in the current frame with respect to *previous frames*, not the ground truth. Thus, when translation and scale and aspect ratio errors are computed with respect to the ground truth, it is expected that SORT will produce inconsistencies and larger errors.

An example to illustrate this point is shown in Figure 2.12, which contains two consecutive samples from the VDD-C test set. The ground truth annotations are presented in black, the predicted locations by YOLOv7 in red, and the predicted locations by YOLOv7+SORT in green. The prediction errors of YOLOv7 in these sequential frames are consistent with respect to the ground truth, resulting in low translation and scale and aspect ratio

errors. On the other hand, the prediction errors of YOLOv7+SORT are consistent with respect to previous frames, resulting in larger translation and scale and aspect ratio errors when compared to the ground truth.

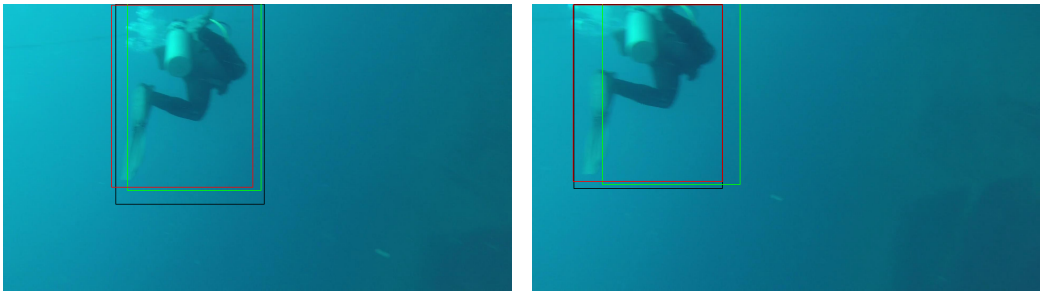


Figure 2.12: Two consecutive frames taken from the VDD-C dataset, highlighting the ability of SORT to smooth detections between frames. The ground truth annotations are presented in black, the YOLOv7 predictions in red, and the YOLOv7+SORT predictions in green.

Frame-to-frame stability

When considering a vision module integrated within a control system, both the stability of information with respect to the ground truth and the stability of information between frames are crucial for effective control. During deployment, the control system relies on stable sequential inputs for accurate motion and trajectory estimation. While metrics such as translation, scale, and aspect ratio error capture stability with respect to the ground truth and provide valuable insights when comparing temporal stability across object detection models, they fail to capture this notion of frame-to-frame stability. As a result, these metrics alone cannot highlight the significance of incorporating a tracking module alongside a detection module for real-world

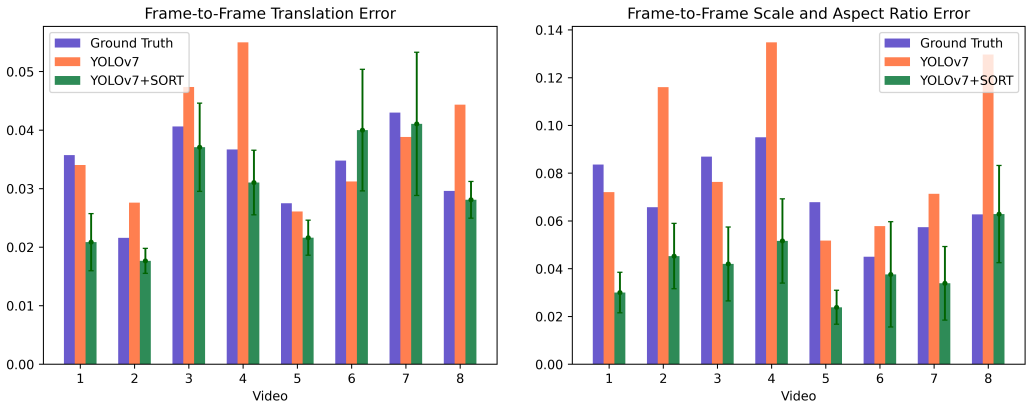
applications.

To measure frame-to-frame temporal stability, we propose an adjustment to both the translation and scale and aspect ratio error definitions. Instead of calculating these errors with respect to the ground truth, we can replace the ground truth with detections from the previous frame. For each video sequence, we calculate the frame-to-frame translation error and the frame-to-frame scale and aspect ratio error using the same calculations presented in Equation 2.9 and Equation 2.6, respectively, but replacing the ground truth with the detection from the previous frame.

Using these metrics, we assess the frame-to-frame temporal stability of YOLOv7+SORT compared to both the standalone YOLOv7 model and the ground truth annotations. Our evaluation is again conducted across each video sequence in the VDD-C test set, and for YOLOv7+SORT, we average the results over 25 combinations of *min_hits* and *max_age* hyperparameters.

The frame-to-frame scale and aspect ratio errors are presented in Figure 2.13b. We observe that YOLOv7+SORT consistently yields the smallest errors across all video sequences. This highlights the ability of SORT to integrate prior predictions with current detections to produce temporally stable outputs that are even smoother than ground truth annotations. In contrast, the standalone YOLOv7 model frequently demonstrates high frame-to-frame scale and aspect ratio errors as it lacks mechanisms to stabilize outputs between frames.

The frame-to-frame translation errors are presented in Figure 2.13a. Sim-



(a) (b)

Figure 2.13: The frame-to-frame translation errors (a), and scale and aspect ratio errors (b), computed for each video sequence in the VDD-C test set. SORT integrated with YOLOv7 exhibits improved frame-to-frame stability by using a Kalman filter to smooth detections between consecutive frames.

ilar to the scale and aspect ratio errors, YOLOv7+SORT consistently yields the lowest errors across all video sequences, with the exception of sequences 6 and 7. We note that these are the same two challenging sequences discussed previously and observe a significant reduction in the average track length of the standalone YOLOv7 model compared to YOLOv7+SORT. This is further supported by Figure 2.10, where YOLOv7 displays notably higher fragmentation errors in these same two sequences. Calculating frame-to-frame metrics over numerous small sequences, as opposed to fewer, larger sequences, can skew the results toward smaller values due to limited data availability for accurate variance estimates.

2.4 Conclusions

In this chapter we presented our vision module, which integrates YOLOv7 for diver detection and SORT for tracking. After evaluating seven state-of-the-art object detection models on the VDD-C dataset, YOLOv7 emerged as the top performer, achieving the highest accuracies while maintaining inference speeds suitable for deployment onboard Aqua's hardware. We explored the benefits and trade-offs of integrating SORT with YOLOv7 and provided guidelines for tuning SORT in practice. During our evaluation of temporal stability, we identified shortcomings in existing metrics, which do not adequately account for practical deployment scenarios where detections need to be stable relative to previous frames, in addition to the ground truth. To address this, we proposed adjustments to these metrics to capture frame-to-frame stability. We demonstrated that SORT can enhance frame-to-frame stability while still maintaining accurate and stable detections when compared against the ground truth.

Looking forward, there are several interesting avenues for future research. In our detection module, we evaluated selected models on the VDD-C dataset in an out-of-the-box fashion with minimal modifications. While this approach was useful for assessing baseline strengths and recent developments, further exploration into custom modifications or incorporating domain-specific knowledge could yield improved results [53, 83]. Future investigations might also explore alternative training schemes, such as transfer learning or training

from scratch, to provide valuable insights into the advantages and disadvantages of each method compared to fine-tuning.

In our tracking module, we plan to refine our proposed frame-to-frame stability metrics by incorporating a weighting mechanism based on tracklet length. We hypothesize that this refinement will offer further insights into the effectiveness of SORT to improve frame-to-frame temporal stability, and emphasize its importance alongside a detection framework. We are also interested in the feasibility of using DeepSORT for real-time tracking. Previous work has shown promising results using DeepSORT for diver identification and re-identification in offline datasets [61]. Building on this, we are curious if this algorithm, with its deep association metric, can be deployed on an AUV to robustly identify and track specific divers.

3

Control module

In a visual servoing system, vision and control are tightly intertwined. Visual inputs directly drive control commands, which shape future visual inputs, and thus future control commands. Designing a robust control module is critical for visual servoing, particularly in challenging marine environments. In this chapter, we explore control paradigms for autonomous underwater vehicles (AUVs) within the context of diver following. We begin by examining traditional methods, specifically Proportional-Integral-Derivative (PID) controllers, which have long been the conventional approach for AUV control. We demonstrate their effectiveness through open ocean experiments and also explore some of their inherent limitations. Following this, we discuss the potential for a reinforcement learning (RL) controller that can encode complex behaviors and autonomously adapt to changes in the system and

environment. Through a series of simulation experiments, we compare the performance of our RL controller with our PID controllers, addressing the strengths and limitations of each approach.

3.1 Background and related work

Designing a control system for an AUV requires careful planning to handle the challenges of dynamic marine environments. In this section, we explore existing paradigms for AUV control systems. We begin by discussing traditional approaches, followed by reinforcement learning and its application to AUV control systems.

3.1.1 Traditional AUV control

Proportional-Integral-Derivative (PID) controllers [10] have long been the conventional approach for AUV control systems [30]. These feedback controllers continuously work to minimize an error signal between a desired setpoint and the current measurement. By considering proportional, integral, and derivative error terms, PID controllers can, when tuned appropriately, apply precise and responsive control outputs to achieve the desired behavior.

PID controllers are popular for AUV control systems due to their relative simplicity, understandability, and ease of implementation. They rely only on the response of the current measurement and can operate without a precise model of the underlying process dynamics. This is particularly beneficial in AUV control systems, which exhibit complex dynamics that can change

when operating in different bodies of water or at varying depths.

Regulated by a set of gains, PID controllers require precise tuning to operate effectively. While popular tuning methods such as the Ziegler-Nichols method [103] exist, tuning these controllers in practice is often a time-consuming and tedious process. Once these gains are set, PID controllers are inherently not adaptable and cannot address changes in the environment or underlying process dynamics without retuning. This can be problematic in constantly changing marine environments and may require manual intervention for long-term missions.

Control systems based on PID control have been implemented in a broad spectrum of AUV platforms. Lensgraf et al. [62] used PID controllers onboard a custom BlueROV2 for precise control in underwater construction tasks. Jung et al. [56] implemented PID controllers for path-following on a fish-like AUV, which moves via actuations to a compliant tail. Fittery et al. [35] used PD controllers onboard a highly maneuverable egg-shaped AUV named the Omni-Egg.

As with other platforms, PID controllers have long been the preferred control paradigm for Aqua. Giguere et al. [39] designed a multi-speed autopilot system for Aqua using PID controllers to achieve desired setpoints. Shkurti et al. [83] used PID controllers in a robot convoy application where two Aqua units visually tracked and followed each other in the ocean. Fulton et al. [37] employed PID controllers to have Aqua autonomously approach scuba divers.

Aside from PID controllers, other traditional AUV control strategies have been explored [101]. Yoerger and Slotine [99] designed a sliding mode controller to deal with nonlinear dynamics and imprecise system models. Smith et al. [84] used a fuzzy logic controller in an autonomous AUV docking algorithm to handle uncertainties in state estimation. Combing both sliding mode and fuzzy logic, Guo et al. [44] developed a sliding mode fuzzy controller for an AUV to maintain a desired heading angle.

3.1.2 Reinforcement learning

Reinforcement learning (RL) algorithms enable robots to *learn* control policies through interaction with their environment [85]. The formal framework for RL is centered around sequential decision-making in a Markov decision process (MDP), a mathematical model that describes the environment in which a robot or agent operates. An MDP is defined by a set of states S, a set of actions A, a transition probability function $P : S \times A \times S \to \mathbb{R}$ that specifies the probability of transitioning from one state to another given an action, and a reward function $r : S \to \mathbb{R}$ that specifies the immediate reward obtained in a given state.

A solution to an MDP can be a deterministic policy $\pi : S \to A$ that maps states to actions, or a stochastic policy $\pi : S \times A \to \mathbb{R}$ that maps states and actions to execution probabilities. The optimal solution to an MDP is a policy π^* that maximizes return, which is some cumulative function of rewards over time.

Environments modeled as MDPs are assumed to obey the Markov property. This property states that the next state and reward depend only on the current state and action, and not on any prior history. While this assumption is helpful when analyzing and developing algorithms to solve MDPs, it does not need to strictly hold to learn an effective control policy in practice.

In RL, the objective is to solve an MDP through interaction with the environment. By exploring the state-action space and receiving feedback via a reward signal, RL algorithms learn and adjust a control policy π to maximize return. Since they learn solely from data and experience, RL algorithms are capable of improvement and adaptation over time.

RL algorithms can be categorized as model-free or model-based. Model-based algorithms such as Probabilistic Inference for Learning Control (PILCO) [24] and Probabilistic Ensembles with Trajectory Sampling (PETS) [18] learn an explicit model of the environment for planning and sequential decision-making. Model-free algorithms, on the other hand, operate without explicit modeling of the environment. They directly learn a policy through interaction and by observing the consequences of actions. Although less sample-efficient than model-based methods [23], model-free methods show promise in complex environments that are challenging to accurately model.

Model-free algorithms can be further divided into value-based and policy-based methods. In value-based RL, collected experience is used to learn a state value function $V(s)$ or a state-action value function $Q(s, a)$. The state value function $V(s)$ represents the expected return of being in state s, while

the state-action value function $Q(s, a)$ represents the expected return of taking action a in state s. Once learned, a control policy can be derived by acting greedily with regards to these value functions. Examples of value-based methods include Q-learning [95] and State–Action–Reward–State–Action (SARSA) [78].

Policy-based methods such as REINFORCE [96] use collected experience to directly learn a control policy, bypassing the need for explicit value function estimation. These methods are beneficial in high-dimensional state and action spaces where it is challenging to feasibly explore and learn a value function. Policy-based methods can also learn stochastic policies, whereas value-based methods are limited to deterministic policies. However, since policy-based methods directly learn a control policy, they tend to exhibit slow convergence times and are prone to getting stuck in local optima. Actor-critic methods, such as Deep Deterministic Policy Gradient (DDPG) [63] and Twin Delayed DDPG (TD3) [36], combine both value-based and policy-based approaches. Figure 3.1 presents a visualization of the family of RL algorithms. Note that this figure is non-exhaustive but provides useful classifications for understanding various approaches in RL.

Tabular RL algorithms represent the policy or value function in a table, explicitly storing values for each state or state-action pair. To scale to high-dimensional or continuous state and action spaces, deep RL leverages neural networks as high-capacity function approximators. Instead of learning a policy or a value function for each state or state-action pair, deep RL

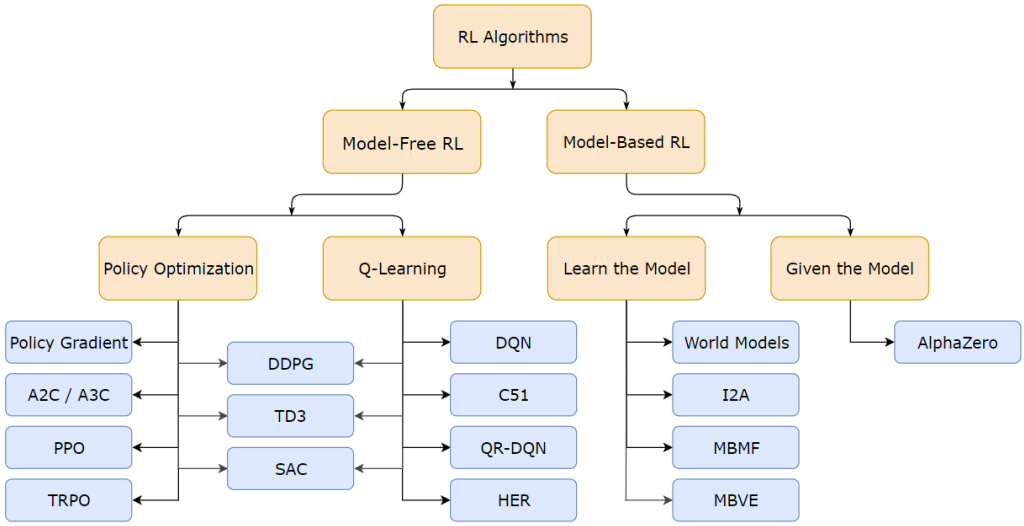


Figure 3.1: A taxonomy of reinforcement learning algorithms. Reprinted from [1].

approximates this using a neural network with weights θ, which are updated based on collected experience. Deep Q-Networks (DQN) [69] and Double DQN (DDQN) [46] are popular examples of value-based deep RL methods.

By using a neural network as a function approximator, deep RL methods can generalize across similar states and actions in the MDP. They can capture intricate patterns and structures within the data, allowing them to learn complex control policies. Although task dependent, deep RL methods often require significant time and computational resources to train. The training process can be unstable and sensitive to hyperparameter choices. Additionally, the integration of a neural network can result in a lack of interpretability of the resulting control policy.

In recent years, RL algorithms have shown impressive results in control

tasks such as robotic manipulation [72], offroad navigation [57], and navigation with underactuated systems [5]. In the underwater domain, the adoption of RL for AUV control has been limited due to challenges such as safely exploring and collecting large amounts of data underwater, and dealing with complex AUV dynamics. Despite these challenges, there have been notable efforts to apply RL and neural networks to AUV control tasks. El-Fakdi and Carreras [32] used an actor-critic algorithm in a vision-based underwater cable inspection system, showcasing results both in simulation and pool environments. Carlucho et al. [13] explored the use of deep RL to output continuous actuator commands for an AUV to maintain a desired reference state.

Specific to Aqua, Meger et al. [68] implemented PILCO to learn closed-loop maneuvers such as U-turns and corkscrews in pool environments. Manderson et al. [67] proposed a neural network controller for a vision-based underwater navigation system that avoids obstacles and explores regions of interest. Rather than reinforcement learning, they followed a behavioral cloning approach for training.

3.2 PID control module

Continuing with a modular visual servoing approach, we directly control the movement of Aqua based on the output from our vision module. As discussed in Chapter 2, our vision module handles the task of diver detection and tracking within the image plane. It outputs bounding box coordinates of

the detected diver, which we use to extract the diver's center coordinates and area. Note that in this work, we do not address multi-diver following and do not explicitly experiment with multiple divers in the field of view (FOV). In cases where our vision module detects two or more divers, we simply select the detection with the highest confidence to feed into our control module. While this is a relatively naive approach, it proved sufficient for our experiments.

Taking the output from our vision module, we normalize the center coordinates in each dimension to the range $[-1, 1]$, representing the relative distance of the diver to the center of the FOV. Similarly, we normalize the area to the range $[0, 1]$, representing the relative size of the detected diver in the FOV. If no diver is detected by our vision module, we consider this a tracking failure and terminate the tracking sequence.

In this section, we outline our PID control module for diver following. We present the results of our open ocean experiments and introduce a simple yet effective mechanism to recover the diver after tracking failures.

3.2.1 Design

When a diver is detected by our vision module, we obtain the normalized center coordinates and area of the detected diver. From these, we define three error signals and employ three separate PID controllers to correct them. The first two controllers receive e_x and e_y, which are the differences between the detected center and the center of the FOV in the x and y dimensions, respectively. These controllers apply yaw and pitch commands to correct the

errors and keep the diver stable in the center of the FOV. An illustration of e_x and e_y is presented in Figure 3.2.

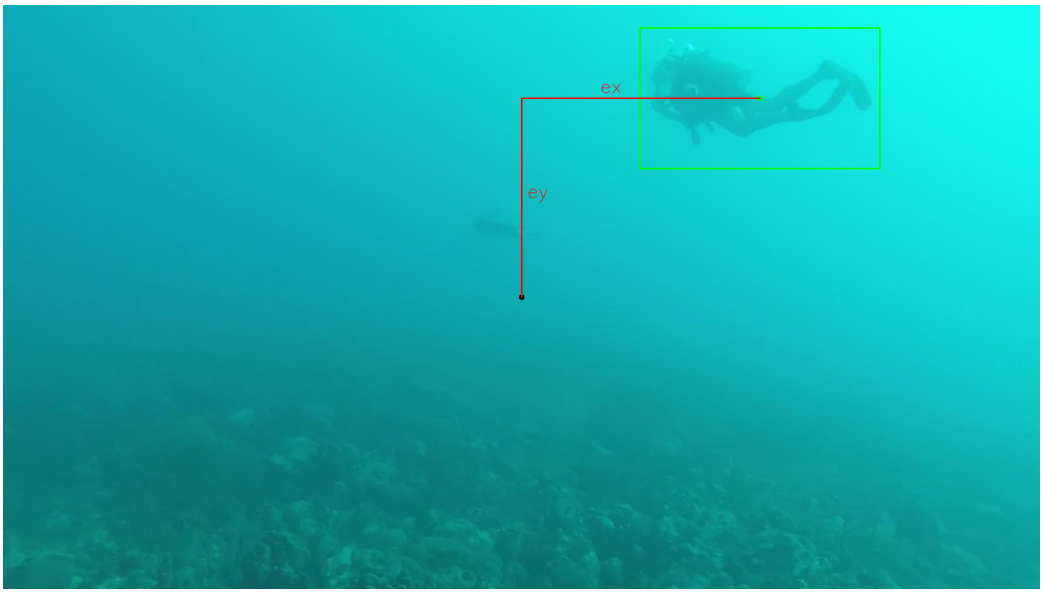


Figure 3.2: Error definitions e_x and e_y used in our PID control module.

The third PID controller receives e_a, which is the difference between the detected area and some target area. It applies linear velocity commands to maintain a preferred distance from the diver. It is worth noting that using the detected area as an indicator of distance can be noisy. Changes in the diver's orientation can cause changes in the detected area without an underlying change in distance. Nevertheless, it provides a weak signal of proximity to avoid crashing into the diver or falling far behind.

The standard PID control law used in our system is defined in Equation 3.1, and a block diagram is presented in Figure 3.3. The resulting control command $u(t)$ at time t for each PID controller is regulated by proportional,

integral, and derivative error terms. The proportional term considers and corrects the current error $e(t)$. The derivative term considers the rate of change of $e(t)$ and outputs commands to avoid overshooting and oscillating around the target. The integral term considers the sum of errors over time and corrects for steady-state errors present in the system.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt} \tag{3.1}$$

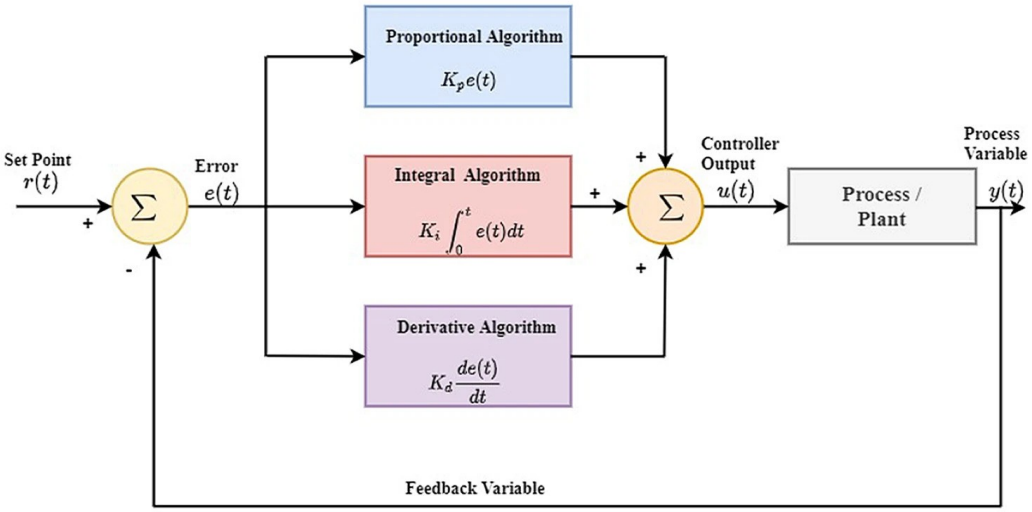


Figure 3.3: A block diagram of a PID control system. The error $e(t)$ represents the difference between the measured process variable $y(t)$ and the desired setpoint $r(t)$. The control output $u(t)$ is determined by the proportional, integral, and derivative error terms. Reprinted from [10].

The proportional, integral, and derivative terms are weighted by gains K_p, K_i, and K_d, respectively. These gains determine how aggressively the

controller responds to each error term, and different weightings can result in significantly different behavior. There are helpful tuning procedures, such as the Ziegler-Nichols method [103], but in practice, tuning these controllers is a time-consuming process and changes to system dynamics will likely necessitate retuning. In our experiments, we spent significant efforts tuning these controllers in the ocean, and system modifications, such as ballasting adjustments, required us to retune.

To avoid variations in the roll angle, which we do not explicitly cover in our visual servoing controller and which could result in unwanted behavior such as flipping over, we integrate a roll stabilizer into our control module. While the pitch, yaw, and linear velocity controllers receive input from our vision module, the roll stabilizer receives the current roll angle reading from an inertial measurement unit (IMU). It follows the same PID control law to stabilize the roll angle at 0°. We tuned this controller in a similar fashion to the others, but it was by far the quickest to tune using consistent feedback from the IMU.

3.2.2 Recovery

To avoid manual resets between tracking sequences, we implement a simple recovery mechanism to handle cases when no diver is detected by our vision module. This mechanism is based on a spiral search algorithm, which searches sequentially in different directions using exponentially increasing excursions away from the starting point [4, 11]. In our application, this cor-

responds to sequentially searching left, right, up, and down for the diver with increasingly longer durations. To enhance this method and quickly recover the lost diver, we initialize the search with the diver's last known location in the image plane. Figure 3.4 shows an example of our recovery mechanism in action.

Although perhaps not the most sophisticated recovery mechanism, it proved remarkably useful in handling failures from the vision module and allowing us to switch to the next tracking sequence without requiring manual intervention. Only when Aqua was clearly lost and unambiguously swimming away from the diver did we manually intervene and reset.

Figure 3.4: An example of our recovery mechanism autonomously recovering a diver after losing visual contact. When our vision module can no longer detect a diver, we employ a spiral search algorithm to sequentially search in different directions for the lost diver. We initialize the search with the diver's last known location for efficient recovery. In this example, the diver was last detected on the left side of the image plane, prompting us to initiate the search by yawing left.

3.2.3 Results

We conducted our open ocean experiments off the west coast of Barbados at the McGill Bellairs Research Institute. These open ocean conditions, with strong currents, provided a challenging environment to test and evaluate our

system. Once tuned, we deployed our PID controllers to autonomously follow a diver using inputs from our vision module. The entire pipeline, from RGB image acquisition to control action, ran at a frequency of 8 Hz onboard Aqua.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Average Duration	236.8	287.6	153.5	273.4	259.5	358.33	271.5	269.16	**425.66**	259.66	206.16	265.33	263.0	208.4
Max Duration	442	417	204	573	359	**783**	451	464	627	451	689	518	668	376

Table 3.1: The average and maximum durations of tracking sequences recorded in 14 open ocean trials. The metric presented is the number of frames.

Our experiments consisted of 68 tracking sequences across 14 trials, comprising a total of 43,710 frames. Table 3.1 highlights the average and maximum durations of the tracking sequences recorded in each trial, reported in frame numbers. Our pipeline achieved an average duration of 267 frames (33.38 seconds), with a maximum of 783 frames (97.88 seconds). Although these numbers may seem modest, it is important to remember our definition of a tracking sequence is relatively strict, terminating after a missed detection for only a single frame. In practice, we observed Aqua successfully following the diver for much longer without intervention, utilizing our recovery mechanism to quickly recover the diver between tracking sequences. Figure 3.5 shows sample frames from a tracking sequence showcasing Aqua robustly following a diver despite dynamic movements.

3.2.4 Limitations

In our experiments discussed in Subsection 3.2.3, our PID controllers demonstrated strong performance, even in challenging open water conditions. This

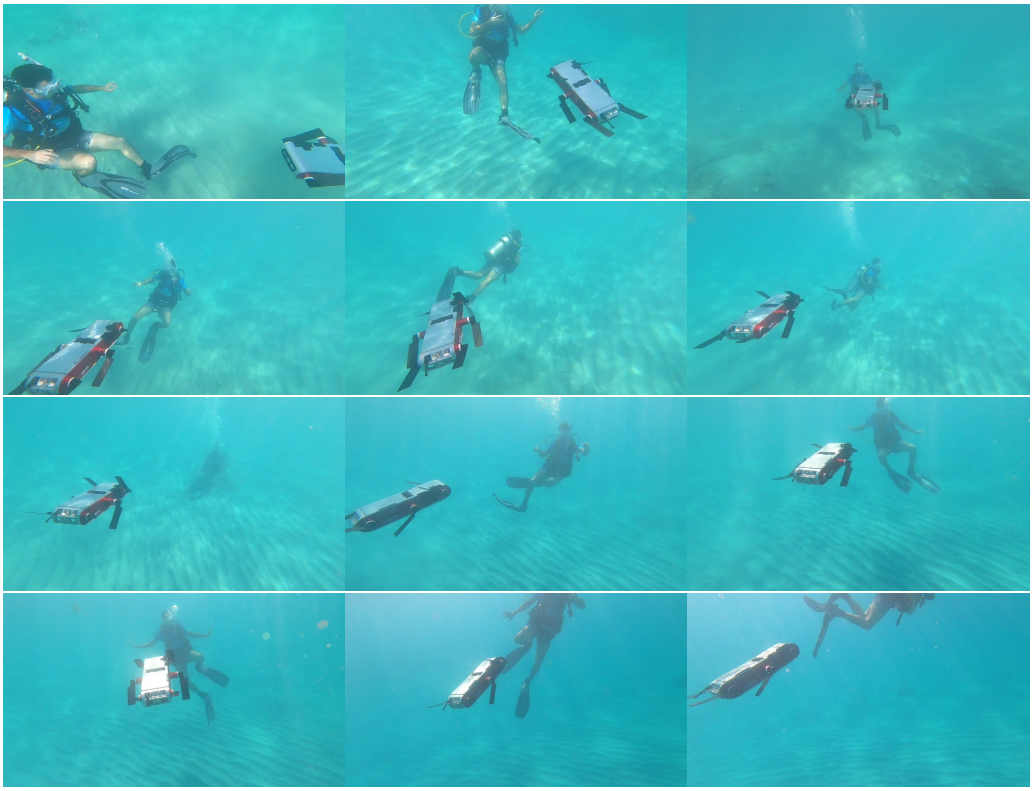


Figure 3.5: Aqua robustly following a diver in the open ocean.

traditional control approach offered simplicity and ease of implementation, while ensuring transparency in the control process. In particular, with our PID controllers, we could precisely calculate the control response based on the diver's location within the FOV. For example, if the diver appeared in the top left of the FOV, we could reliably anticipate that Aqua would yaw left and pitch up. Similarly, if the diver swam far away, we could be certain that Aqua would increase its linear velocity to close the distance.

While simplicity and understandability in the control system are desirable for underwater human-robot collaboration, our PID controllers suffer from

three limitations. First, they require precise manual tuning, which proved to be a time-consuming and tedious process. They are not self-adaptable, necessitating frequent retuning due to changes in ballasting or environmental conditions. For example, on days with strong currents, we had to adjust the controllers to deliver more aggressive responses, whereas on calm days, we had to revert to a more neutral setting.

Second, PID controllers lack the capability to learn and coordinate with the strengths and weaknesses of our vision module. In scenarios where our vision module performed poorly in regions of the image plane due to factors like sand or debris on the camera lens, our PID controllers lacked a mechanism to compensate. They struggled to adapt and keep the diver centered in regions where our vision module excelled.

Third, by employing separate controllers for yaw, pitch, and linear velocity, our PID controllers fail to address interdependencies between these control variables. In general, applying pitch rates causes changes to the pitch angle, and applying yaw rates causes changes to the yaw angle. However, due to the complex dynamics of Aqua, these control commands are not always independent. Notably, when the roll angle deviates from zero, pitch rates affect the yaw angle, and yaw rates affect the pitch angle. These effects become more pronounced with larger roll deviations and when traveling at higher speeds.

Although we incorporated a roll stabilizer to mitigate these interdependencies, deploying in open water conditions with strong currents often re-

sulted in deviations from the ideal roll angle of 0°. In these situations, our separated PID controllers, responsible for independently applying pitch and yaw rates, were constantly in conflict. While these conflicts were not significant enough to prevent our controllers from effectively following the diver, employing separated PID controllers in these cases is a suboptimal control approach, potentially increasing energy consumption and the likelihood of failures.

3.3 Reinforcement learning control module

Inspired by the shortcomings of our PID control module described in Subsection 3.2.4, in this section we investigate the use of an RL-based controller for diver following. We first frame the problem as an MDP, carefully designing components such as the state space, action space, and reward signal. We then employ Double Deep Q-Networks (DDQN) to solve the MDP and learn a control policy. We present results from our simulated environment and compare them with our PID control module. We also explore whether our learning-based controller can address some of the shortcomings of our PID controllers, namely coordinating with our vision module and adapting to changing environmental conditions.

3.3.1 Diver following as a Markov decision process

To develop our learning-based controller, we first formulate the diver following problem as an MDP. This requires defining components such as the state

space S, action space A, reward function $r : S \to \mathbb{R}$, initial state, and termination conditions. For the transition probability function $P : S \times A \times S \to \mathbb{R}$, we assume it is provided by the environment and can only be sampled through interaction.

State space

Receiving the output from our vision module, we define the state vector at time t as $s_t = [x_t, y_t, a_t, d_t]$. Here, $[x_t, y_t, a_t]$ represents the normalized diver center coordinates and area, while d_t is a binary flag indicating the presence or absence of a diver in the current frame. This binary flag is included for scenarios where our vision module fails to detect a diver in the FOV. In such cases, the state vector includes the last known diver state and an indicator that no diver was detected, giving Aqua the required information to recover after a tracking failure.

Providing Aqua solely with information about the current state vector s_t presents a significant challenge due to the diver following task being partially observable. Aqua cannot fully observe the internal state of the environment, but can only observe the state vector s_t provided by our vision module. This state vector alone lacks vital details concerning diver movement, Aqua's movement, and external disturbances, leading to many internal states in the system being perceptually aliased. This adds a layer of complexity, as the information provided by s_t is insufficient to comprehensively understand the situation and make optimal decisions.

For example, if the diver is located on the left side of the FOV, the optimal control action could differ significantly based on the diver's velocity. Another example is if Aqua is yawing left yet the diver remains on the left side of the FOV. This situation could be due to the diver swimming left at an equal velocity, or a current pushing Aqua in the opposing direction. To address this, we augment s_t with a history of state vectors and previously taken actions, a commonly used technique in reinforcement learning algorithms [69, 73]. Equation 3.2 outlines our augmented state vector, s_t^+, with a history of size h. This historical context reduces the number of perceptually aliased states and provides Aqua with the required information to learn the consequences of its actions and make optimal decisions. In our experiments we use a history size of 10.

$$s_t^+ = [s_t, a_{t-1}, s_{t-1}, a_{t-2}, ..., a_{t-h+1}, s_{t-h+1}] \tag{3.2}$$

Action space

We illustrate our proposed neural network architecture in Figure 3.6. The augmented state vector is input to a sequence of three shared fully connected layers, each followed by a Rectified Linear Unit (ReLU) activation function. Inspired by Manderson et al. [67], we adopt a three-headed architecture where each head predicts state-action values for discretized pitch, yaw, or linear velocity rates. Each head considers seven evenly spaced rates within specified

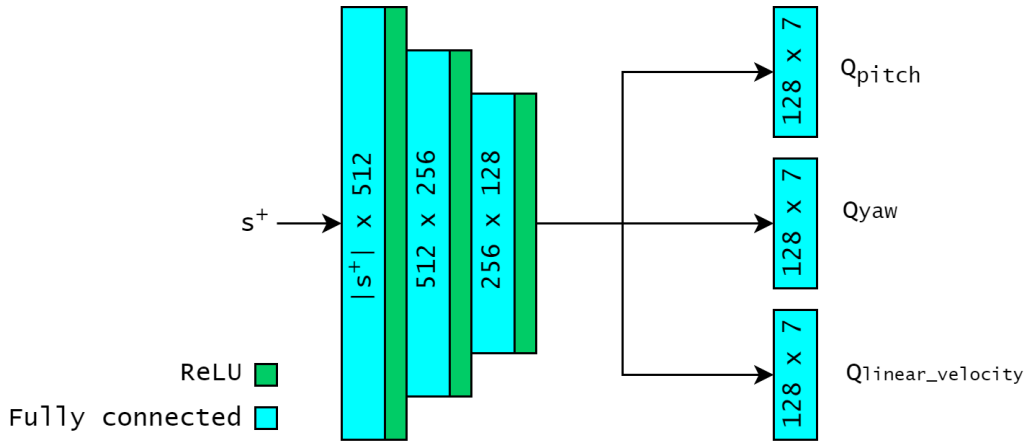


Figure 3.6: Our three-headed neural network architecture for diver following. The network includes three shared fully connected layers, each followed by a ReLU activation function. Each output head is responsible for predicting Q-values for discretized pitch, yaw, or linear velocity rates. The input to our network, s_t^+, represents our augmented state vector, which encompasses a history of diver detections and previously taken actions.

ranges: $[-0.075, 0.075]$ for pitch, $[-0.75, 0.75]$ for yaw, and $[0.25, 1.0]$ for linear velocity. Our initial experiments involving diver following via manual control revealed that these seven rates per head were sufficient for robust tracking while maintaining a manageable action set for a machine learning model to learn.

A multi-headed architecture with discretized actions offers several advantages over single-headed or continuous action space architectures. Discretizing the action space and limiting the number of possible actions reduces problem complexity and enhances the data efficiency of our learning pipeline. Additionally, by partitioning pitch, yaw, and linear velocity rates into separate heads, we further simplify the action space, as each head only needs

to consider actions within a single control dimension. This is in contrast to a single-headed architecture, which would need to handle all possible combinations of the selected actions. This design choice facilitates scalability, allowing additional rates to be considered without causing the action space to explode.

In our design process, we also considered three fully separated networks. In this approach, each network would receive only the relevant information from the state vector and output control commands in a single dimension. For instance, the pitch network would receive a history of y center coordinates and previously taken pitch rates, and would be responsible for only predicting pitch state-action values. While this isolated approach might converge faster by omitting the shared layers and further breaking down the problem, it cannot handle the interdependencies between control variables, which was a core motivation for exploring beyond traditional separated PID controllers. By including shared layers, which comprise the majority of the network, our multi-headed architecture can learn interactions between control variables and manage them to maximize a common reward signal.

However, our multi-headed architecture with discretized actions has some limitations. By only considering a finite number of rates, we potentially lose the precise control that a continuous action space could provide. Additionally, a single-headed architecture may be better equipped to handle interdependencies among control variables. While our multi-headed architecture can manage these interdependencies through shared layers and optimization

to maximize a common reward signal, a single-headed architecture that outputs pitch, yaw, and linear velocity rates may deliver superior performance at the cost of having a larger action space to explore.

Reward signal

Reward signal design plays a pivotal role in the learning dynamics of RL. Poorly crafted signals can result in inefficient learning and suboptimal control policies. Sparse reward signals, which are infrequently presented to the robot or agent during training, are often easy to define and allow the agent full flexibility to learn and discover novel strategies. Sparse rewards can be temporally sparse, spatially sparse, or both.

Spatially sparse reward signals are presented only at specific states within the environment. For example, in the context of diver following, a spatially sparse reward signal might be given only when the diver is located in the center of the FOV. Temporally sparse reward functions are presented at infrequent periods based on the duration of the task. In diver following, a temporally sparse reward signal could be achieved only after successfully following the diver for a given duration.

Sparse reward signals provide the agent with limited guidance, often requiring longer training horizons and more data to converge to an effective control policy. In contrast, dense reward signals are presented to the agent frequently throughout the learning process, offering rich intermediate feedback allowing the agent to quickly learn the consequences of its actions. De-

signing dense reward signals can be challenging, especially for applications where the value of intermediate states is ambiguous. Dense rewards may also inadvertently constrain exploration, steering the agent towards policies that align closely with the dense reward signal and potentially inhibiting the discovery of novel strategies.

In our application of diver following, the primary objective is to maximize the duration the diver remains in the FOV. This naturally suggests a sparse reward definition where the agent is simply rewarded a constant value when the diver is present in the FOV. While this reward signal holds promise, it overlooks the rich spatial information available in the state vector. The agent receives the same reward whether the diver is detected at the center or the edge of the FOV, disregarding the fact that a diver near the edge is more susceptible to being lost. This reward signal also does not encode a measure of proximity, and the agent would receive the same reward regardless of the detected area of the diver. The agent would have to learn these spatial relationships, which could significantly prolong training time.

Instead, we propose a dense reward signal r_t outlined in Equation 3.3, where Aqua is rewarded based on the diver's proximity to the center of the FOV and the target area which we set to 0.02. If a diver is detected, the reward is computed using a Gaussian distribution with $\sigma_1 = 0.5$ pixels based on the distance between the diver and the center of the FOV. The reward is scaled based on the difference between the detected area and the target area, again using a Gaussian distribution with $\sigma_2 = 0.025$. We omit

the normalization factors to ensure rewards remain within the range $[0, 1]$, regardless of the chosen sigma values. If no diver is detected, the agent receives a penalty of -1.

$$r_t(s_t^+) = \begin{cases} \exp\left(-\frac{x_t^2+y_t^2}{2\sigma_1^2}\right) \times \exp\left(-\frac{(a_t-0.02)^2}{2\sigma_2^2}\right) & \text{if } d_t = 1, \\ -1 & \text{if } d_t = 0 \end{cases} \tag{3.3}$$

Figure 3.7 shows slices of our reward signal at various detected areas. By considering both x and y coordinates in the distance calculation, the resulting reward signal exhibits radial symmetry in the image plane. By scaling based on the detected area, the reward signal also encodes a measure of preferred distance. This design choice facilitates a common reward signal for pitch, yaw, and linear velocity actions, enabling Aqua to learn interactions between control variables and manage them to keep the diver stable in the FOV while maintaining proximity.

Initial and terminal states

We adopt an episodic approach for training, treating each tracking sequence as a distinct episode. To initialize, we position the simulated diver directly in front of Aqua, aligning with the center of its FOV. Aqua then swims forward at a constant rate until the augmented state vector is populated before applying pitch, yaw, and linear velocity commands. We model diver movement with random velocity changes every 5 seconds to reflect the frequent

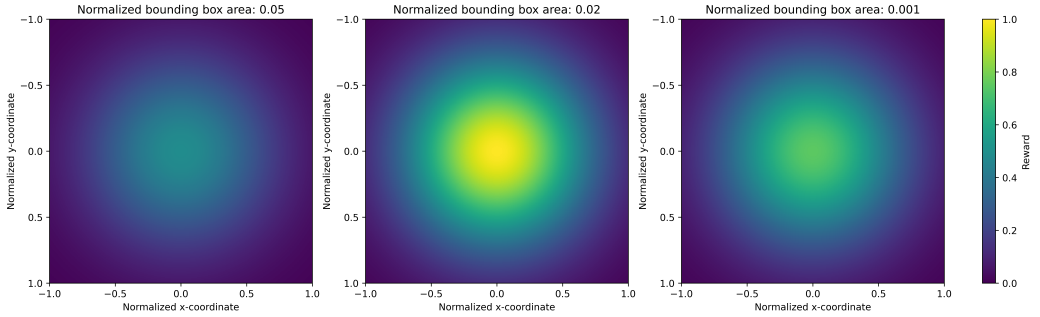


Figure 3.7: Slices of our reward signal in the image plane when Aqua is close to the diver (left), maintaining the preferred distance from the diver (center), and lagging behind the diver (right).

changes in movement divers make throughout a dive. Additionally, we bound the diver's maximum speed to ensure Aqua can reliably follow the diver if appropriate control commands are selected.

We consider two terminal states for an episode. The first occurs when the vision module does not detect the diver for 25 consecutive frames. In this case, we assume the diver is lost and cannot be recovered, and we end the episode in failure. The second occurs when the tracking sequence exceeds 3000 frames. In this scenario, Aqua has successfully followed the diver for over 5 minutes, and we end the episode in success. We found that our framework is not sensitive to the choice of these two hyperparameters; thus, they can be adjusted to place emphasis on recovery or long-term tracking.

3.3.2 Training

To solve our formulated MDP and learn a control policy for diver following, we employ Double Deep Q-Networks (DDQN) [46]. DDQN is a value-based

deep RL algorithm that estimates the optimal state-action value function $Q^*(s_t, a_t)$ using a neural network with weights θ. This function $Q^*(s_t, a_t)$ represents the maximum expected return after taking action a_t in state s_t, following *any* control policy. In our problem, we define return as the discounted sum of rewards over an infinite horizon $\sum_{t'=t}^{\infty} \gamma^{t'-t} r_t$, where γ is set to 0.9. It is important to note that although we end episodes and reset the environment after 3000 frames, Aqua is trained to maximize expected return over an infinite horizon. Following Pardo et al. [70], we use the maximum duration solely to run multiple trials and experiments, without allowing Aqua to experience transitions causing an environmental reset due to reaching the maximum duration.

The optimal state-action value function $Q^*(s_t, a_t)$ obeys the Bellman equation, shown in Equation 3.4. This recursive equation states that $Q^*(s_t, a_t)$ is equal to the immediate reward obtained plus the maximum discounted state-action value over possible actions in the next state s_{t+1}. Following this equation, we can define a loss function based on the difference between the current estimate $Q(s_t, a_t; \theta)$ and the target $r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$. We can then iteratively train a neural network to minimize this loss using collected experience.

$$Q^*(s_t, a_t) = r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \tag{3.4}$$

The DDQN algorithm builds on this idea by including an identical but slower-moving target network with weights θ^- to stabilize training and prevent overestimation of Q-values. The target computation used in DDQN is defined in Equation 3.5.

$$r_{t+1} + \gamma Q(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a'; \theta); \theta^-) \tag{3.5}$$

In our training procedure, we use our architecture in Figure 3.6 to define an ϵ-greedy policy network with ϵ decreasing exponentially from 0.9 to 0.1 over the first 250,000 timesteps. At each timestep t, Aqua receives s_t^+ from our vision module, saves a transition $(s_{t-1}^+, a_{t-1}, s_t^+, r_t)$ to a list of collected experiences, takes an action according to the policy network, then performs one optimization step and one soft-update step. An optimization step involves randomly sampling a batch of transitions, computing the Smooth L1 loss [41] between the current Q-values and the DDQN target, then backpropagating the loss through the policy network. The soft-update step, defined in Equation 3.6, moves the target network's weights θ^- closer to the policy network's weights θ at a rate regulated by τ. In our experiments, we set $\tau = 0.0025$ to ensure our target network is slow-moving. This helps maintain consistent target values during training, promoting stability and preventing oscillations in Q-values.

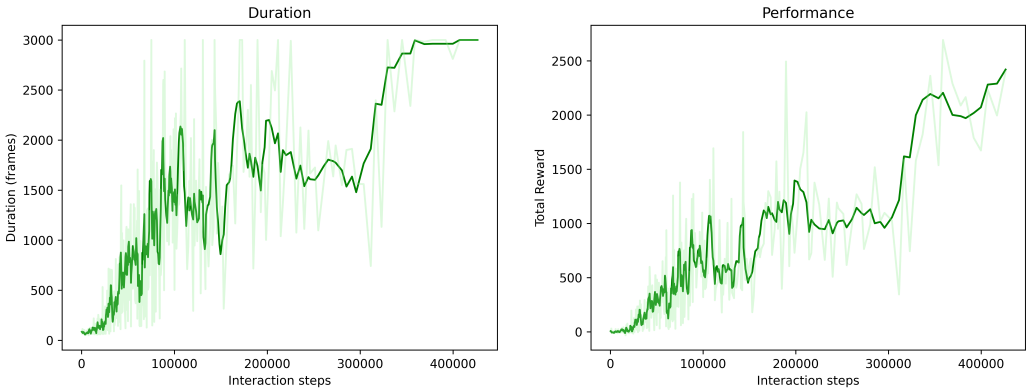
$$\theta^- = \tau\theta + (1-\tau)\theta^- \tag{3.6}$$

To periodically evaluate the performance of our system, after every five training episodes with the ϵ-greedy policy network, we conduct an evaluation episode where actions are greedily selected from the target network. The resulting control policy can be expressed as $\pi(s_t^+) = \arg\max_a Q(s_t^+, a; \theta^-)$ for each head. We record the total reward obtained during the episode and the duration in frames. Figure 3.8 illustrates how Aqua learns to successfully follow the diver in one of our experiments through interactions with its environment.

To ensure our simulation pipeline mimics real-world scenarios, we reduce the frequency of incoming images to our vision module to 10 Hz. With the processing time of both our vision and control modules, our simulation pipeline, from RGB image to control action, operates at a speed of approximately 8 Hz, matching our real-world implementation. Note that we do not make any changes to the rate at which the simulation environment updates; we simply try to mimic the processing speed when our system is deployed onboard Aqua.

3.3.3 Results

Once our target network with weights θ^- converges on an estimation of the optimal state-action value function, we conduct our evaluation studies. Dur-



(a) (b)

Figure 3.8: An example of how our DDQN control policy learns to follow a diver through interactions with the environment. After every five training episodes, we conduct an evaluation episode where actions are greedily selected from the target network. We present the evaluation durations (a) and total rewards (b) throughout the training process. The results are smoothed using a moving average with a window size of five episodes.

ing these studies, we freeze the weights and select actions greedily from the target network over 10 evaluation episodes. We record both the episode duration and the total reward obtained.

We compare the performance of our trained DDQN control policy with our PID control policy, which we precisely tune for our simulation environment. Recall that our PID controllers output continuous pitch, yaw, and linear velocity rates, while our DDQN control policy is restricted to discretized rates. We use the same termination conditions for both policies: terminating after no detections for 25 consecutive frames or upon reaching the maximum duration of 3000 frames.

	Duration	Total Reward
PID	**3000.0 ± 0.0**	**2903.2 ± 8.4**
DDQN	2912.4 ± 205.8	2491.2 ± 294.5

Table 3.2: A comparison between our DDQN and PID control policies. During evaluation, we record the duration in frames and total reward obtained for each policy over 10 episodes, reporting the mean and standard deviation. While both policies demonstrate strong performance, our well-tuned PID control policy delivers superior results in diver following.

Table 3.2 displays the duration and total reward obtained for both our DDQN and PID control policies, averaged over 10 evaluation episodes. From these results, we observe that both policies can effectively follow the diver, yet well-tuned PID controllers are capable of delivering superior performance. This is reinforced by Figure 3.9a and Figure 3.9b, which show how our PID controllers can maintain a preferred distance from the diver and precisely keep it stable in the center of the FOV. This enables our PID control policy to consistently reach the maximum duration in each of the evaluation episodes and obtain rewards close to the maximum in each frame.

Our DDQN control policy can also effectively follow the diver, but it cannot achieve the same consistency and stability as our PID controllers. As shown in Figure 3.9c and Figure 3.9d, our DDQN control policy can maintain a preferred distance and keep the diver in the center of the FOV, but with higher variance. Additionally, in one of the evaluation episodes, our DDQN policy takes an action that causes the diver to be lost before reaching the maximum duration. Dealing with a policy dictated by a neural network, we have limited visibility on why this particular behavior occurs and the reasons

behind this modest performance. Possible factors include our reward signal design, our choice to discretize the action space, and our decision to use a multi-headed architecture.

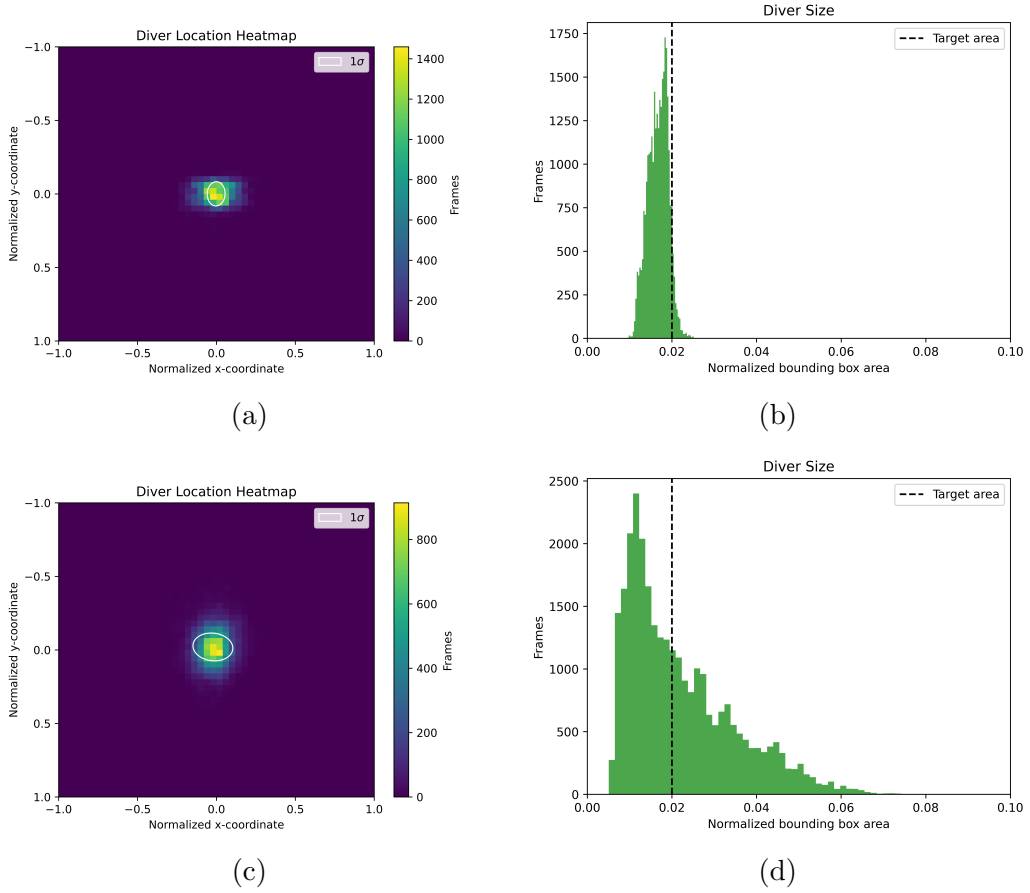


Figure 3.9: An analysis of our PID and DDQN control policies across 10 evaluation episodes. Among all the detected diver locations and areas, our PID control policy effectively keeps the diver stable in the center of the FOV (a) and maintains the desired distance (b). Our DDQN control policy also keeps the diver stable (c) and maintains the desired distance (d), but with higher variance.

Coordinating with the vision module

In Subsection 3.3.3, we compared our DDQN control policy with our PID control policy, which we precisely tuned for our simulation environment. While this comparison highlighted the strengths of well-tuned PID controllers and reinforced their widespread adoption for AUV control, a more intriguing study is to explore how these control policies adapt to changes in the system and environment.

When deploying in constantly changing underwater environments, robust and adaptive control systems are essential. PID controllers, while effective, are inherently not adaptable and require manual tuning or modifications to adjust to new conditions. In contrast, one of the core benefits of an RL-based controller is its ability to learn and self-adapt from experience. This adaptability makes RL-based controllers a promising direction for marine applications where environmental conditions can change rapidly.

To investigate this adaptability, we conduct two studies where we introduce variations in the system and environment. We assess how both control policies respond to these changes, focusing on their ability to maintain effective diver following under altered conditions.

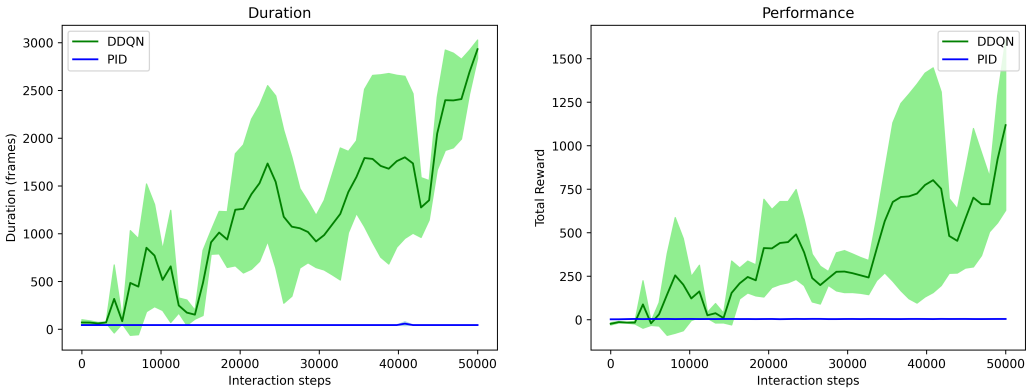
In the first study, we recreate a practical scenario observed during our real-world deployments where debris partially obscures the camera lens, making it difficult to detect divers in those regions. As shown in Figure 3.10, we intentionally blur a large section of the FOV, causing our vision module to

Figure 3.10: To explore the adaptability of our control policies, we intentionally blur a large region of the image plane, making it extremely difficult for our vision module to detect a scuba diver in that area.

perform poorly in that area.

Consistent with the study in Subsection 3.3.3, we compare our trained DDQN control policy with our PID control policy, recording key metrics such as episode duration and total reward. However, in this study, we allow online optimization, meaning we permit the weights of the DDQN control policy to be updated based on collected experience. No other changes, such as modifications to the reward signal or diver movement patterns, were made.

We conduct three experiments starting from the same trained DDQN weights and investigate how our DDQN and PID control policies adapt to our weakened vision module. Figure 3.11 presents the results of this study, including the mean and standard deviation across the three experiments. Initially, we observe poor performance from both policies as they attempt



(a) (b)

Figure 3.11: A comparison of how our DDQN and PID control policies adapt to a weakness in our vision module. Initially, both policies struggle to follow the diver. However, through interaction with the environment, our DDQN control policy learns the weakness and adjusts to effectively follow the diver. Our PID controllers cannot adapt without manual intervention. We perform three experiments starting with the same initialization and present the mean and standard deviation of the episode durations (a) and total rewards (b).

to keep the diver centered in the image plane, where it is more susceptible to becoming lost. However, through interactions with the environment, the DDQN control policy learns this complex situation and shifts the diver towards areas of the image plane where the vision module can reliably detect it. It understands that the diver can no longer be reliably detected in the center of the image plane, and adjusts by settling for decreased rewards on the left side of the image plane as shown in Figure 3.12. Our PID control policy, on the other hand, lacks a mechanism to adapt. It naively tries to keep the diver centered in the FOV, where the vision module struggles to detect it.

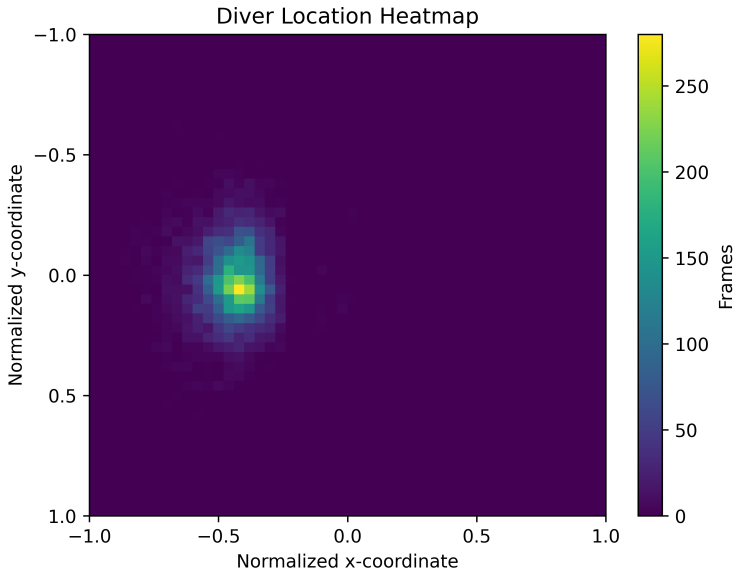


Figure 3.12: Diver location distribution from an evaluation episode with our DDQN control policy. The policy understands that our vision module is weaker on the right side of the FOV and takes actions to keep the diver stable on the left side, despite receiving smaller rewards.

Adapting to changes in the environment

In this second study, rather than investigating how our control policies adapt to weaknesses in our vision module, we explore their adaptability to changes in the surrounding environment. Specifically, we drastically reduce the water density in our simulation environment, causing control actions to become much more aggressive. This scenario simulates real-word deployments in different bodies of water, where the same control command can have drastically different effects. Ideally, a robust control system should be capable of adapting to these changes without requiring extensive remodeling or retuning.

Similar to our weakened vision module study, we conduct three exper-

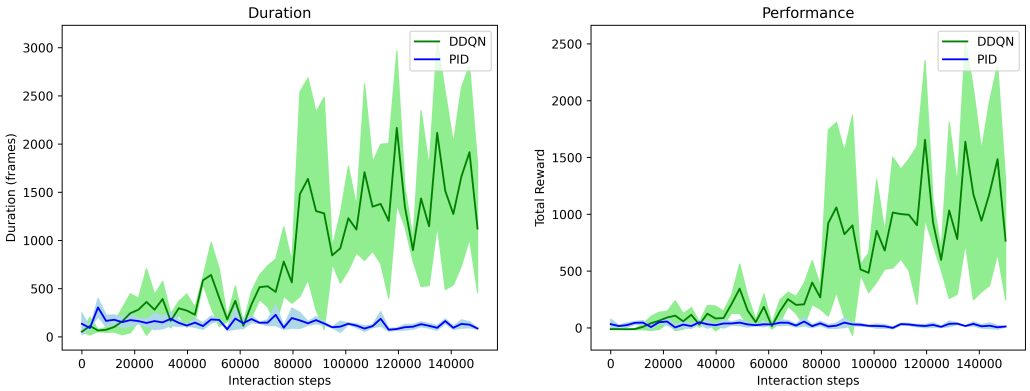
iments starting with our trained DDQN weights and investigate how our DDQN and PID control policies adapt to this environmental change. We permit our DDQN control policy to update its weights based on newly collected experience but do not make any other modifications to the problem setup.

The results of this study are presented in Figure 3.13. We see initially, both policies struggle due to the control actions becoming much more aggressive. However, through interactions with the environment, our DDQN control policy learns this dynamics change and consistently avoids taking aggressive actions which would cause the diver to quickly become lost. Our static PID controllers have no mechanism to adapt and would likely need significant manual retuning to handle this change in water density.

3.3.4 Limitations

In our experiments discussed in Subsection 3.3.3, our DDQN control policy demonstrated strong performance and the ability to adapt to complex changes in both the system and the surrounding environment. While these properties are desirable for robust AUV control systems, our DDQN control policy suffers from two major limitations.

First, using a control policy dictated by a neural network offers limited visibility into the underlying control process. This lack of transparency makes it difficult to predict future control actions and provides limited guidance when suboptimal actions are taken. With our PID control policy, we can



(a) (b)

Figure 3.13: A comparison of how our DDQN and PID control policies adapt to a reduction in water density. Both policies initially struggle to follow the diver due to control actions becoming much more aggressive. However, through interaction, our DDQN control policy learns this dynamics change and consistently chooses low magnitude actions to keep the diver stable in the FOV. Our PID controllers cannot adapt without manual retuning. We perform three experiments starting from the same initialization and present the mean and standard deviation of episode durations (a) and total rewards (b) for both policies.

be certain of the resulting control action given the diver's state, ensuring predictable and understandable behavior. Our DDQN control policy lacks these guarantees, which can be problematic in real-world scenarios where transparency in the control system is vital for human-robot collaboration.

Second, our DDQN control policy requires a substantial amount of data and experience to learn an effective control strategy. Despite discretizing the action space and using a multi-headed architecture, training from scratch demands approximately 400,000 interaction steps to converge, equating to almost 13 hours of interaction with the environment at 8 Hz. In challenging

underwater environments, this process is likely to take much longer, and acquiring such data is expensive, time-consuming, and potentially dangerous. Additionally, during the training process, the nature of our DDQN control policy necessitates exploration, often resulting in the diver becoming lost as it learns to avoid these situations in the future. This can be undesirable, especially when PID controllers can quickly provide solutions for each part of the state space without an extensive need for exploration.

Focusing on adaptability, while our DDQN control policy can autonomously adapt, it still requires time and experience to do so. For instance, in both the weakened vision module and reduced water density experiments, our DDQN control policy needed over 50,000 interaction steps to adapt, equating to almost 2 hours of interaction with the environment at 8 Hz. This introduces a tradeoff between retuning PID controllers for out-of-distribution scenarios and allowing an RL-based controller to self-adapt through trial and error. Each approach has its own set of challenges and time requirements.

3.4 Conclusions

In this chapter we explored the design and implementation of two AUV control paradigms within the context of diver following. In particular, we analyzed traditional, widely adopted PID controllers and an RL-based controller trained using Double Deep Q-Networks (DDQN).

Our open ocean experiments revealed that PID controllers, while simple and easy to implement, can be highly effective when appropriately tuned.

They produce predictable and responsive control actions, without requiring a model of the underlying process dynamics. However, PID controllers require precise manual tuning, and are limited to minimizing a concrete error signal. They cannot handle interdependencies between control variables, and lack the ability to adapt to changes in the system or environment. These limitations can pose significant challenges, especially for long-term deployments in marine settings.

Through extensive simulation experiments, our RL-based controller demonstrated a remarkable ability to autonomously adapt to complex changes in the system and environment. Adaptability is a desirable property when deploying in marine settings where conditions can drastically change. Our RL-based controller can also learn and manage interactions between control variables through shared network layers and optimization to maximize a common reward signal. Despite these advantages, our RL-based controller requires substantial data and experience to learn an effective control policy, which can be a limiting factor for practical deployments. The inherent lack of transparency in neural network-based control policies also raises concerns regarding predictability and stability.

Balancing the immediate effectiveness and transparency of PID controllers with the learning capabilities and adaptability of RL-based controllers remains an open challenge. In future research, we would like to focus on hybrid approaches that combine PID control for initial deployments and safety with the adaptability of RL for long-term improvement. Additionally, we

would like to focus on methods to improve the efficiency and robustness of our RL-based control policy for practical deployments. This could involve extensive hyperparameter tuning, modifications to the MDP model, or alternative learning schemes. It could also involve exploring Sim2Real techniques, which leverage simulation environments to develop robust policies that can be transferred to real-world applications [71, 89].

4

Discussion and future work

Underwater human-robot collaboration is a promising area within the robotics community. Given their advanced sensing capabilities, autonomous underwater vehicles (AUVs) working alongside scuba divers can significantly enhance underwater safety and tasks related to oceanic monitoring and mapping. However, for AUVs to operate effectively as scuba diver companions, they must be capable of reliably tracking and following scuba divers. In this thesis, we tackled the problem of diver following with an AUV from two key perspectives: perception and control. For perception, we developed a vision-based module to autonomously detect and track divers within the image plane. For control, we investigated both traditional and deep learning-based paradigms for diver following in challenging marine environments. Notably, our framework operates without requiring a system dynamics model, which

is often difficult to obtain for complex AUV systems. In this final chapter, we review our findings and discuss their broader implications. We evaluate the strengths and limitations of our approach and identify avenues for future research in underwater autonomy.

4.1 Discussion

Inspired by the effectiveness and versatility of vision-based sensing, we adopted a visual servoing approach for diver following, directly controlling Aqua's movement using feedback from an onboard camera. We chose to modularize our system into distinct vision and control modules. This allowed us to modify each module without significant changes to the other and provided a general framework that can easily extend to track and follow other underwater targets. However, this modular approach also introduced a strong dependency between our vision and control modules, necessitating concrete steps to ensure robustness on both ends of the problem. In this section, we review our proposed system for diver following, discussing both our vision and control modules.

4.1.1 Vision module

In Chapter 2, we introduced our vision module for diver detection and tracking. For diver detection, we fine-tuned seven deep learning-based object detectors on the Video Diver Detection (VDD-C) dataset, evaluating both accuracy and inference speed. Among these detectors, You Only Look Once

version 7 (YOLOv7) achieved the highest accuracy and the second-highest inference speed, making it the preferred choice for integration into our vision module.

To conduct mirrored experiments in simulation, we collected the Simulated Diver Detection (SDD) dataset, consisting of 1,603 images of a simulated diver. We analyzed the diversity of this dataset and enhanced it through data augmentation techniques. Upon evaluating YOLOv7 on the SDD dataset, we found that the diver detection task in simulation was easier than in real-world conditions. While this result was not surprising, it motivated us to explore techniques to handle cases of missing and false detections, and also to improve the temporal stability of our vision module.

For diver tracking, we integrated the Simple Online and Realtime Tracking (SORT) algorithm alongside YOLOv7. We demonstrated the effect of SORT on detection accuracy through an ablation study on two hyperparameters that help fill in missed detections and filter out false detections. From this study, we found that detection accuracy was not significantly reduced by using SORT to smooth detections between frames; in fact, for certain hyperparameter combinations, SORT improved detection accuracy by leveraging information from previous frames. Based on these findings, we provided guidelines for tuning SORT in practical applications.

When examining temporal stability, we evaluated the impact of SORT on key stability metrics such as fragmentation error, scale and aspect ratio error, and translation error. We identified a shortcoming in existing metrics, as

they fail to address frame-to-frame stability, where detections need to remain consistent relative to previous frames, rather than solely to the ground truth. We proposed adjustments to these metrics to better capture frame-to-frame stability and demonstrated that SORT significantly enhances frame-to-frame stability while maintaining strong detection accuracy.

The integration of YOLOv7 with SORT in our vision module presents several notable strengths and limitations. By utilizing explicit mechanisms for object detection and tracking, our vision module leverages the state-of-the-art detection capabilities of YOLOv7 and enhances them with an explicit tracking algorithm for temporal stability. One of the key strengths of this approach is flexibility; we can easily update the module, for example, by upgrading to the latest YOLOv9 or exploring alternative tracking algorithms, without significant changes to the overall pipeline. The combination of a single-shot object detector with a lightweight tracker also enables our vision module to achieve inference speeds suitable for real-time deployment. Unlike previous work that primarily focused on offline datasets [98], our vision module is carefully designed for computational efficiency, ensuring that both components are lightweight and practical for deployment on embedded systems.

Despite these strengths, our vision module also has certain limitations. Apart from fine-tuning on the VDD-C dataset, we did not make any domain-specific modifications to YOLOv7, relying instead on its out-of-the-box capabilities for diver detection. While this allowed us to maintain a general

problem formulation, it is likely that custom modifications or incorporating domain-specific knowledge would yield improved results. Considering our multi-object tracker, SORT does not consider object appearance during its data association step. Although this helps our module achieve fast processing times, it also means that it struggles to differentiate between two closely located divers. In this thesis, we did not conduct experiments to track multiple divers and thus did not directly encounter this issue, but it is likely that future extensions of this work will require a more robust mechanism to address this limitation.

4.1.2 Control module

In Chapter 3, we explored two paradigms for AUV control within the context of diver following. First, we investigated Proportional-Integral-Derivative (PID) controllers, a conventional approach for AUV control systems. We designed three separate PID controllers to output pitch, yaw, and linear velocity commands solely from visual input. Additionally, we developed a PID controller to stabilize Aqua's roll angle using an inertial measurement unit (IMU) and a spiral search mechanism to recover a scuba diver after losing visual contact. We deployed both our vision and PID control modules in the ocean off the west coast of Barbados, achieving a frequency of 8 Hz from image acquisition to control output. Across 68 tracking sequences, our system successfully followed a diver for an average duration of 267 frames (33.38 seconds) with a maximum of 783 frames (97.88 seconds). Our recovery

mechanism was also able to quickly recover the diver between failures without requiring manual intervention.

Despite the success of our PID control module, it suffered from three significant limitations. First, it required precise manual tuning, which proved to be a time-consuming and sensitive process. Second, it lacked self-adaptability, requiring manual adjustments to handle changes in system dynamics or the surrounding environment. Third, it had no mechanism to handle the interdependencies between control variables, resulting in controllers that often conflicted with each other during deployments.

To address these limitations and explore the potential of learning-based control, we implemented a control policy for diver following using reinforcement learning (RL). We started by framing the diver following problem as a Markov decision process (MDP), carefully designing components such as the state space, action space, reward signal, and network architecture. We then employed Double Deep Q-Networks (DDQN) to solve the MDP and learn a control policy.

Through simulation experiments, we demonstrated that well-tuned PID controllers could outperform our RL-based controller in the task of diver following. However, in terms of adaptability and generalizability, our RL-based controller proved superior. It autonomously adapted and learned to handle changes in the system and environment. Specifically, in scenarios where debris obstructed part of the camera lens or the water density changed, our RL-based controller adapted successfully while our PID controllers could

not without manual intervention.

One significant limitation of our RL-based control module is its need for a substantial amount of data to learn a control policy and adapt to changes. This can be problematic for real-world deployment, where it can be expensive, time-consuming, and dangerous to acquire such extensive data. A notable trade-off emerges between retuning PID controllers for out-of-distribution settings and allowing an RL-based controller to self-adapt through interaction.

Another significant limitation arises from our RL-based control policy being dictated by a neural network and offering limited transparency in the underlying control process. While our PID control module allowed precise calculation of the control output given the diver's state, our RL-based control module lacked such guarantees. This made it difficult to understand when seemingly suboptimal control commands were selected, raising concerns for stability and effectiveness in underwater human-robot collaboration.

Beyond the inherent limitations of our proposed control paradigms, there are three notable areas for improvement in this work. First, our assumption that a tracking sequence ends after a single missed detection is unnecessarily strict. In practice, Aqua followed the diver for much longer periods, switching between successful tracking and autonomous recovery. This behavior aligns more naturally with how scuba divers follow each other in the open ocean, frequently checking on their partner rather than continuously keeping them in view. Relaxing the assumption that a tracking sequence ends after a

single missed detection and instead measuring the duration without manual intervention, or quantifying the balance between tracking and recovery, would have been beneficial.

Second, throughout this thesis, we discussed the inability of separate PID controllers to handle the interdependencies between control variables, and how our RL-based controller can potentially manage them through shared network layers and a common reward signal. While we examined this concept from a theoretical and optimal control standpoint, we did not conduct experiments to specifically evaluate this trade-off. It remains unclear whether our RL-based controller truly learned to manage these interdependencies or if these interdependencies were significant enough to hinder the performance of our PID controllers.

Third, experiments regarding our RL-based controller were conducted solely in simulation. While these simulations were helpful in exploring the benefits and potential of learning-based control, reinforcing these ideas with practical deployments would have been ideal. Our initial attempts to deploy our RL-based controller in practice faced significant barriers such as the need for extensive data and limited transparency in the control process. During our initial deployments, Aqua frequently lost the diver when using our DDQN control policy. It was unclear whether this behavior resulted from a lack of data and experience, instability in the training process, an issue with our problem design, or Aqua converging to suboptimal behavior.

4.2 Future work

In addition to directly addressing the limitations of this work discussed in Section 4.1, there are two intriguing directions for future work. The first is to focus on methods to safely and effectively deploy an RL-based controller in practice. While this thesis has primarily explored whether an RL-based controller *can* learn to follow a diver and adapt to system and environmental changes, it has not extensively addressed how quickly or efficiently it can achieve this.

One approach to facilitate the deployment of RL-based control is to adopt a shared control scheme. In this scheme, tuned PID controllers would be used to safely explore the state-action space and stabilize the training process. For example, in diver following, PID controllers could be activated when the diver is located in the outer regions of the image plane to ensure stability and minimize failures. The RL-based controller would be activated in the inner parts of the image plane, where it could explore suboptimal actions without losing visual contact with the diver. As training progresses, this inner region could be gradually expanded, allowing the RL-based controller to take control over larger parts of the image plane. Our initial simulation experiments with this approach revealed that while overall training time was not significantly reduced, the number of failures and resets drastically decreased. Thus, this approach could be a promising direction for safely learning control policies in practice with reduced manual intervention.

Another promising direction to leverage the benefits of RL while mitigating the data-scarcity problem in the real world is to transfer a control policy learned in simulation to the real world. Large amounts of training data can be collected in simulation at a relatively low cost. The learned policy can then be transferred in a zero-shot approach, where it is directly used for evaluation, or in a few-shot approach, where it serves as a robust starting point, requiring much less data in the real world to achieve suitable performance.

However, control policies learned in simulation often fail to generalize to the real world due to the large distribution shift between the two environments. Physics in simulation are only approximations of the real world, and accurately simulating real-world dynamics is challenging. To successfully transfer a control policy learned in simulation to the real world, steps must be taken to ensure the policy is robust to modeling errors in both the system dynamics and the environment.

Domain Randomization is a common technique used to bridge the distribution gap between simulation and reality [89]. By injecting random variations and disturbances into the simulated environment, the agent is exposed to a wide range of variations during training, which helps prevent overfitting and enables the policy to generalize well to different evaluation environments. Robust Adversarial Reinforcement Learning (RARL) is another popular technique for learning robust control policies [71]. This model-agnostic technique involves adding an adversarial agent that strategically applies disturbances during training. Through this approach, the adversary agent learns to op-

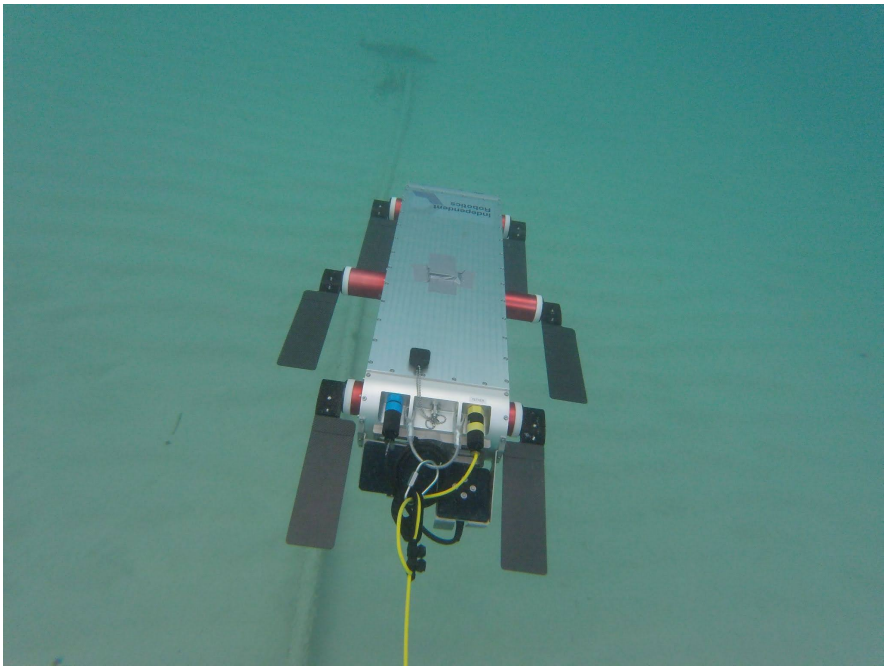


Figure 4.1: An extension of our work to autonomously inspect a pipeline on the seafloor using visual feedback.

timally apply disturbances, while the protagonist learns to accomplish the specified task despite these disturbances. The resulting policies have been shown to be robust to parameter variations and environmental changes during evaluation. Investigating these methods further for the task of diver following could be beneficial in developing a control policy capable of being transferred to the real world.

The second interesting direction is to generalize our framework to other underwater visual servoing tasks. For example, one direction we are currently pursuing, shown in Figure 4.1, is autonomously inspecting a subsea pipeline with Aqua. While there are differences in the nuances of the problem, the

general theme of controlling Aqua's movement from visual input remains consistent. Instead of using a front-facing camera, we can use a downward-facing camera to detect and segment the pipeline in the image plane. We can then take control actions to progress along the pipeline for inspection.

By building on this work and exploring these two directions, we can enhance the robustness and efficiency of AUVs in visual servoing applications. This will work towards advancing methods for underwater human-robot collaboration and autonomy.

Publications

- Lotfi, F., **Virji, K.**, and Dudek, G. (2023) Robust scuba diver tracking and recovery in open water using YOLOv7, SORT, and spiral search. In *2023 20th Conference on Robots and Vision (CRV)*, pages 233-240.
- Lotfi, F., **Virji, K.**, Faraji, F., Berry, L., Holliday, A., Meger D., and Dudek, G. (2024) Uncertainty-aware hybrid paradigm of nonlinear MPC and model-based RL for offroad navigation: Exploration of transformers in the predictive model. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*.

Acronyms

- **AP**: **A**verage **P**recision
- **AUV**: **A**utonomous **U**nderwater **V**ehicle
- **CNN**: **C**onvolutional **N**eural **N**etwork
- **DDQN**: **D**ouble **D**eep **Q**-**N**etworks
- **DETR**: **DE**tection **TR**ansformer
- **FOV**: **F**ield **O**f **V**iew
- **FPS**: **F**rames **P**er **S**econd
- **GPU**: **G**raphics **P**rocessing **U**nit
- **IMU**: **I**nertial **M**easurement **U**nit
- **IoU**: **I**ntersection **o**ver **U**nion
- **MDP**: **M**arkov **D**ecision **P**rocess
- **PID**: **P**roportional-**I**ntegral-**D**erivative
- **RL**: **R**einforcement **L**earning
- **SDD**: **S**imulated **D**iver **D**etection
- **SORT**: **S**imple **O**nline and **R**ealtime **T**racking
- **VDD-C**: **V**ideo **D**iver **D**etection
- **YOLO**: **Y**ou **O**nly **L**ook **O**nce

Bibliography

[1] Achiam, J. (2018). Spinning up in deep reinforcement learning. Retrieved from https://spinningup.openai.com/en/latest/.

[2] Agarwal, T., Fulton, M., and Sattar, J. (2021). Predicting the future motion of divers for enhanced underwater human-robot collaboration. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5379–5386.

[3] Aharon, N., Orfaig, R., and Bobrovsky, B. (2022). BoT-SORT: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*.

[4] Baezayates, R., Culberson, J., and Rawlins, G. (1993). Searching in the plane. *Information and Computation*, 106(2):234–252.

[5] Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82.

[6] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. S. (2016). Fully-convolutional siamese networks for object tracking. In Hua, G. and Jégou, H., editors, *Computer Vision – ECCV 2016 Workshops*, pages 850–865, Cham. Springer International Publishing.

[7] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468.

[8] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

[9] Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550.

[10] Borase, R. P., Maghade, D. K., Sondkar, S. Y., and Pawar, S. N. (2021). A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2):818–827.

[11] Burlington, S. and Dudek, G. (1999). Spiral search as an efficient mobile robotic search technique. Technical report, Center for Intelligent Machines, McGill University.

[12] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham. Springer International Publishing.

[13] Carlucho, I., De Paula, M., Wang, S., Petillot, Y., and Acosta, G. G. (2018). Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning. *Robotics and Autonomous Systems*, 107:71–86.

[14] Chavez, A. G., Pfingsthorn, M., Birk, A., Rendulić, I., and Misković, N. (2015). Visual diver detection using multi-descriptor nearest-class-mean random forests in the context of underwater human robot interaction (HRI). In *OCEANS 2015 - Genova*, pages 1–7.

[15] Chen, S., Ma, K., and Zheng, Y. (2019). Med3D: Transfer learning for 3D medical image analysis.

[16] Chen, X., Yu, J., and Wu, Z. (2020). Temporally identity-aware SSD with attentional LSTM. *IEEE Transactions on Cybernetics*, 50(6):2674–2686.

[17] Choi, W. (2015). Near-online multi-target tracking with aggregated local flow descriptor. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3029–3037.

[18] Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[19] Codd-Downey, R. and Jenkin, M. (2019a). Finding divers with SCUBANet. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5746–5751.

[20] Codd-Downey, R. and Jenkin, M. (2019b). Human robot interaction using diver hand signals. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 550–551.

[21] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.

[22] Danelljan, M., Häger, G., Khan, F. S., and Felsberg, M. (2017). Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575.

[23] Deisenroth, M. P., Neumann, G., and Peters, J. (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):1–142.

[24] Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: a model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 465–472, Madison, WI, USA. Omnipress.

[25] DeMarco, K. J., West, M. E., and Howard, A. M. (2013). Sonar-based detection and tracking of a diver for underwater human-robot interaction scenarios. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2378–2383.

[26] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[27] Dicle, C., Camps, O. I., and Sznaier, M. (2013). The way they move: Tracking multiple targets with similar appearance. In *2013 IEEE International Conference on Computer Vision*, pages 2304–2311.

[28] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[29] Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H. (2023). StrongSORT: Make DeepSORT great again. *IEEE Transactions on Multimedia*, 25:8725–8737.

[30] Dudek, G. and Jenkin, M. (2010). *Computational Principles of Mobile Robotics*. Cambridge University Press, 2nd edition.

[31] Dudek, G., Jenkin, M., Prahacs, C., Hogue, A., Sattar, J., Giguere, P., German, A., Liu, H., Saunderson, S., Ripsman, A., Simhon, S., Torres, L.-A., Milios, E., Zhang, P., and Rekletis, I. (2005). A visually guided swimming robot. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3604–3609.

[32] El-Fakdi, A. and Carreras, M. (2013). Two-step gradient-based reinforcement learning for underwater robotics behavior learning. *Robotics and Autonomous Systems*, 61(3):271–282.

[33] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL visual object classes challenge 2007 (VOC2007) results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[34] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL visual object classes challenge 2012 (VOC2012) results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[35] Fittery, A., Mazumdar, A., Lozano, M., and Asada, H. H. (2012). Omni-egg: A smooth, spheroidal, appendage free underwater robot capable of 5 dof motions. In *2012 Oceans*, pages 1–5.

[36] Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

[37] Fulton, M., Hong, J., and Sattar, J. (2022). Using monocular vision and human body priors for AUVs to autonomously approach divers. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1076–1082.

[38] Georgiades, C., German, A., Hogue, A., Liu, H., Prahacs, C., Ripsman, A., Sim, R., Torres, L.-A., Zhang, P., Buehler, M., Dudek, G., Jenkin, M., and Milios, E. (2004). Aqua: an aquatic walking robot. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 4, pages 3525–3531 vol.4.

[39] Giguere, P., Girdhar, Y., and Dudek, G. (2013). Wide-speed autopilot system for a swimming hexapod robot. In *2013 International Conference on Computer and Robot Vision*, pages 9–15.

[40] Giguere, P., Prahacs, C., and Dudek, G. (2006). Characterization and modeling of rotational responses for an oscillating foil underwater robot. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3000–3005.

[41] Girshick, R. (2015). Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448.

[42] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158.

[43] Gomez Chavez, A., Ranieri, A., Chiarella, D., Zereik, E., Babić, A., and Birk, A. (2019). CADDY underwater stereo-vision dataset for human–robot interaction (HRI) in the context of diver activities. *Journal of Marine Science and Engineering*, 7(1).

[44] Guo, J., Chiu, F.-C., and Huang, C.-C. (2003). Design of a sliding mode fuzzy controller for the guidance and control of an autonomous underwater vehicle. *Ocean Engineering*, 30(16):2137–2155.

[45] Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., and Wang, S. (2017). Learning dynamic siamese network for visual object tracking. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1781–1789.

[46] Hasselt, H. v., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press.

[47] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[48] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2015). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(03):583–596.

[49] Heo, B., Yun, S., Han, D., Chun, S., Choe, J., and Oh, S. J. (2021). Rethinking spatial dimensions of vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11916–11925.

[50] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[51] Hutchinson, S., Hager, G., and Corke, P. (1996). A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670.

[52] Independent Robotics (2024). Aquasim - the underwater robot simulator [computer software]. Retrieved from https://www.independentrobotics.com/robot-simulator.

[53] Islam, M. J., Fulton, M., and Sattar, J. (2019). Toward a generic diver-following algorithm: Balancing robustness and efficiency in deep visual detection. *IEEE Robotics and Automation Letters*, 4(1):113–120.

[54] Islam, M. J. and Sattar, J. (2017). Mixed-domain biological motion tracking for underwater human-robot interaction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4457–4464.

[55] Jocher, G. (2020). YOLOv5 by ultralytics (version 7.0) [computer software]. Retrieved from https://github.com/ultralytics/yolov5.

[56] Jung, D. S., Pott, P. P., Salumäe, T., and Kruusmaa, M. (2013). Flow-aided path following of an underwater robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 4602–4607.

[57] Kahn, G., Abbeel, P., and Levine, S. (2021). BADGR: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2):1312–1319.

[58] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

[59] Kvasić, I., Mišković, N., and Vukić, Z. (2019). Convolutional neural network architectures for sonar-based diver detection and tracking. In *OCEANS 2019 - Marseille*, pages 1–6.

[60] Langis, K. d., Fulton, M., and Sattar, J. (2021). Towards robust visual diver detection onboard autonomous underwater robots: Assessing the effects of models and data. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5372–5378.

[61] Langis, K. d. and Sattar, J. (2020). Realtime multi-diver tracking and re-identification for underwater human-robot collaboration. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11140–11146.

[62] Lensgraf, S., Balkcom, D., and Li, A. Q. (2023). Buoyancy enabled autonomous underwater construction with cement blocks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5207–5213.

[63] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[64] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

[65] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham. Springer International Publishing.

[66] Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2.

[67] Manderson, T., Gamboa Higuera, J. C., Wapnick, S., Tremblay, J.-F., Shkurti, F., Meger, D., and Dudek, G. (2020). Vision-based goal-conditioned policies for underwater navigation in the presence of obstacles. *Robotics: Science and Systems XVI*.

[68] Meger, D., Higuera, J. C. G., Xu, A., Giguère, P., and Dudek, G. (2015). Learning legged swimming gaits from experience. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2332–2338.

[69] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[70] Pardo, F., Tavakoli, A., Levdik, V., and Kormushev, P. (2018). Time limits in reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4045–4054. PMLR.

[71] Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. (2017). Robust adversarial reinforcement learning. In Precup, D. and Teh, Y. W., editors,

Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, pages 2817–2826. PMLR.

[72] Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. (2018). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

[73] Ramstedt, S. and Pal, C. (2019). Real-time reinforcement learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[74] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

[75] Redmon, J. and Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525.

[76] Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

[77] Richardson, D., editor (2010). *PADI Open Water Diver Manual*. PADI, Rancho Santa Margarita, CA, 2nd edition.

[78] Rummery, G. A. and Niranjan, M. (1994). On-line q-learning using connectionist systems. *Technical Report, University of Cambridge Department of Engineering*.

[79] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[80] Sattar, J. and Dudek, G. (2007). Where is your dive buddy: tracking humans underwater using spatio-temporal features. In *2007 IEEE/RSJ*

International Conference on Intelligent Robots and Systems, pages 3654–3659.

[81] Sattar, J. and Dudek, G. (2018). Visual identification of biological motion for underwater human–robot interaction. *Autonomous Robots*, 42(1):111–124.

[82] Sattar, J., Giguere, P., Dudek, G., and Prahacs, C. (2005). A visual servoing system for an aquatic swimming robot. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1483–1488.

[83] Shkurti, F., Chang, W.-D., Henderson, P., Islam, M. J., Higuera, J. C. G., Li, J., Manderson, T., Xu, A., Dudek, G., and Sattar, J. (2017). Underwater multi-robot convoying using visual tracking by detection. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4189–4196.

[84] Smith, S., Rae, G., Anderson, D., and Shein, A. (1993). Fuzzy logic control of an autonomous underwater vehicle. *IFAC Proceedings Volumes*, 26(1):318–323. 1st IFAC International Workshop on Intelligent Autonomous Vehicles, Hampshire, UK, 18-21 April.

[85] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press, 2nd edition.

[86] Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312.

[87] Tao, R., Gavves, E., and Smeulders, A. M. (2016). Siamese instance search for tracking. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1420–1429, Los Alamitos, CA, USA. IEEE Computer Society.

[88] Terven, J., Córdova-Esparza, D.-M., and Romero-González, J.-A. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716.

[89] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

[90] Unity Technologies (2024). Unity engine (version 2020.3.31) [computer software]. Retrieved from https://unity.com/products/unity-engine.

[91] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[92] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I.

[93] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7464–7475.

[94] Wang, C.-Y., Yeh, I.-H., and Liao, H.-Y. M. (2024). YOLOv9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616*.

[95] Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[96] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

[97] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649.

[98] Xia, Y. and Sattar, J. (2019). Visual diver recognition for underwater human-robot collaboration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6839–6845.

[99] Yoerger, D. and Slotine, J. (1985). Robust trajectory control of underwater vehicles. *IEEE Journal of Oceanic Engineering*, 10(4):462–470.

[100] Yu, F., Li, W., Li, Q., Liu, Y., Shi, X., and Yan, J. (2016). POI: Multiple object tracking with high performance detection and appearance feature. In Hua, G. and Jégou, H., editors, *Computer Vision – ECCV 2016 Workshops*, pages 36–42, Cham. Springer International Publishing.

[101] Yuh, J. (2000). Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, 8(1):7–24.

[102] Zhang, H. and Wang, N. (2017). On the stability of video detection and tracking. *arXiv preprint arXiv:1611.06467*.

[103] Ziegler, J. G. and Nichols, N. B. (1993). Optimum settings for automatic controllers. *Journal of Dynamic Systems, Measurement, and Control*, 115(2B):220–222.

[104] Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276.