TU Delft

Delft University of Technology

Document Version
Final published version

Citation (APA)
Ilioudi, A. (2025). *Deep - learning - based computer vision for underwater environments*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:047e28e3-2b91-4d60-a851-920ca716f287

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

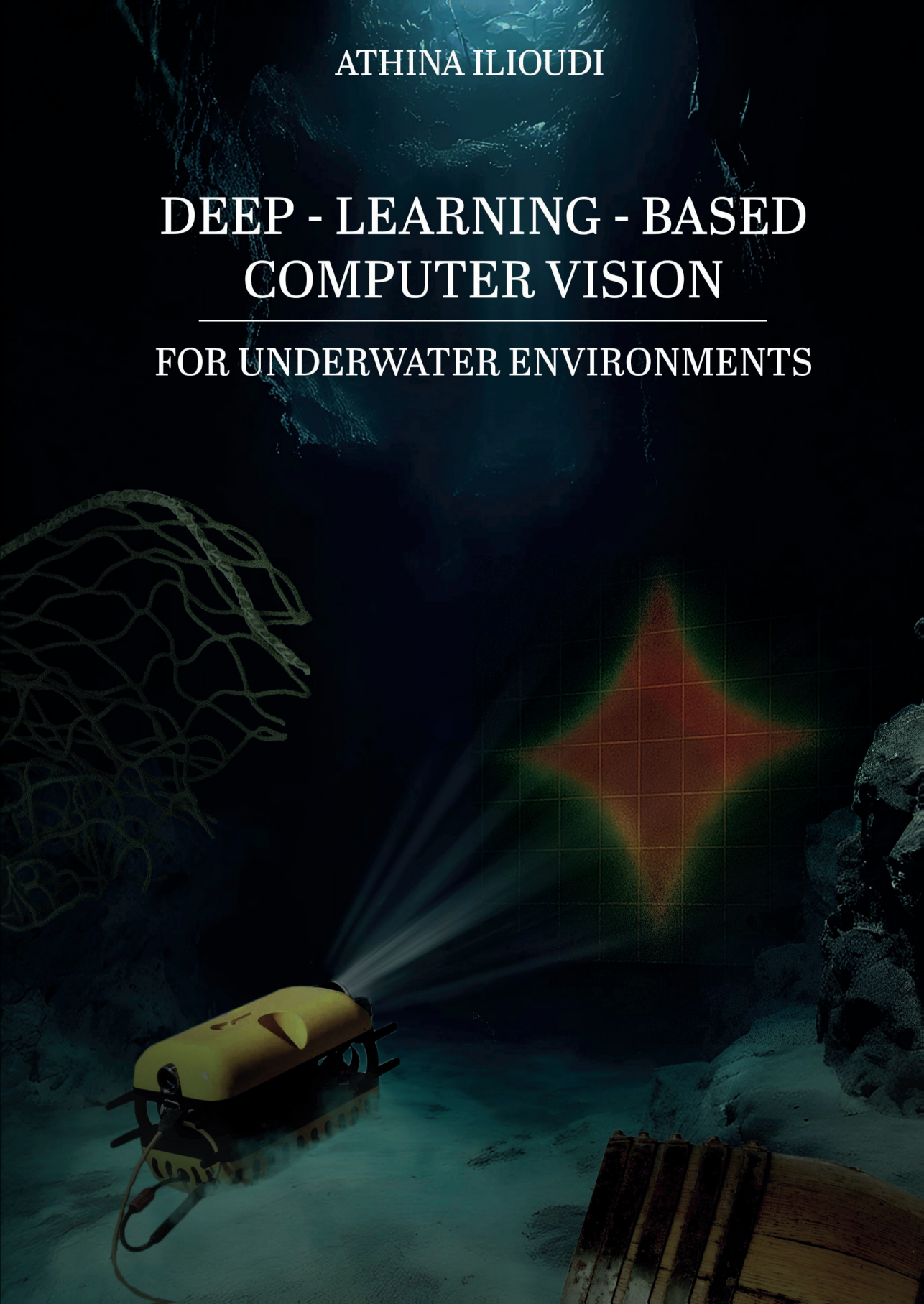
Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

ATHINA ILIOUDI

DEEP - LEARNING - BASED COMPUTER VISION

FOR UNDERWATER ENVIRONMENTS



DEEP-LEARNING-BASED COMPUTER VISION FOR UNDERWATER ENVIRONMENTS

DEEP-LEARNING-BASED COMPUTER VISION FOR UNDERWATER ENVIRONMENTS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Wednesday, June 04, 2025 at 17:30

by

Athina ILIOUDI

Master of Electrical Engineering,
Eindhoven University of Technology, Netherlands,
born in Larissa, Greece.

This dissertation has been approved by the promotors. Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. B. De Schutter,	Delft University of Technology, *promotor*
Dr. A. Dabiri,	Delft University of Technology, *copromotor*
Independent members:	
Prof. dr. ir. M.J.T. Reinders,	Delft University of Technology
Prof. dr. R. Babuska,	Delft University of Technology
Dr. M.A. Valdenegro-Toro,	University of Groningen
Prof. dr. I.L. Busoniu,	Technical University of Cluj-Napoca, Romania
Dr. J. Sattar,	University of Minnesota, USA





This work was supported by SeaClear, European Union's Horizon 2020 Research and Innovation Programme under Grant 871295.

ISBN 978-94-93431-61-4

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

CONTENTS

Summary ix

Samenvatting xi

1 Introduction 1
- 1.1 Background . . . 2
- 1.2 Motivation and challenges . . . 2
- 1.3 Research questions . . . 3
- 1.4 Contributions . . . 4
- 1.5 Thesis outline . . . 5

2 Deep Learning for Object Detection and Segmentation in Videos: Toward an Integration with Domain Knowledge 7
- 2.1 Introduction . . . 8
- 2.2 Preliminaries . . . 8
  - 2.2.1 Computer vision tasks . . . 9
  - 2.2.2 Deep learning vs. traditional computer vision techniques . . . 10
  - 2.2.3 Image-based deep learning methods . . . 11
- 2.3 Deep learning methods for object detection and segmentation in videos . . . 14
  - 2.3.1 Optical flow . . . 15
  - 2.3.2 Tracking . . . 17
  - 2.3.3 Long short-term memory . . . 19
  - 2.3.4 Gated recurrent unit . . . 20
  - 2.3.5 Self-attention mechanism . . . 22
  - 2.3.6 Generative learning . . . 23
- 2.4 Challenges in deep-learning-based computer vision . . . 25
- 2.5 Integrating deep learning with domain knowledge . . . 26
  - 2.5.1 Motivation . . . 26
  - 2.5.2 Hybrid architectures . . . 26
  - 2.5.3 Hybrid architecture implementation in computer vision . . . 30
- 2.6 Future directions in deep learning for object detection and segmentation in videos . . . 30
- 2.7 Conclusions . . . 31

3 A Dataset for Detection of Underwater Marine Debris 33
- 3.1 Background & summary . . . 34
- 3.2 Methods . . . 35
  - 3.2.1 Data acquisition . . . 37
  - 3.2.2 Underwater image quality assessment . . . 38

3.2.3 Fusion based image enhancement . . . 39
3.3 Data records . . . 40
3.4 Technical validation . . . 43
3.4.1 Baseline models . . . 44
3.4.2 Experimental setup . . . 46
3.4.3 Baseline results . . . 46

4 The SeaClear System: An Intelligent Multi-Robot Solution for Autonomous Cleanup of Marine Debris on the Seabed 55
4.1 Introduction . . . 56
4.2 Related work . . . 57
4.3 SeaClear system description . . . 59
4.3.1 Robotic modules . . . 59
4.4 Software architecture for system integration . . . 62
4.5 Proposed system workflow & methodology . . . 63
4.5.1 Overview . . . 63
4.5.2 Pose estimation . . . 65
4.5.3 Aerial and underwater mapping . . . 66
4.5.4 Litter collection . . . 73
4.5.5 Coordination between UAV and USV during operation . . . 76
4.6 Experimental results and analysis . . . 77
4.6.1 Simulation model . . . 77
4.6.2 Field experiments . . . 80
4.7 Conclusions and future work . . . 89
Appendix . . . 90
4.A Field of view requirements . . . 90
4.B Technical specifications . . . 90

5 Towards Establishing an Automated Selection Framework for Underwater Image Enhancement Methods 95
5.1 Introduction . . . 96
5.2 Preliminaries . . . 97
5.2.1 Image quality assessment metrics . . . 97
5.2.2 Underwater image enhancement methods . . . 98
5.3 Proposed selection scheme . . . 98
5.4 Methodology . . . 98
5.4.1 Dataset synthesis . . . 99
5.4.2 Convolutional neural network training . . . 100
5.4.3 Selection framework . . . 101
5.5 Experimental results . . . 102
5.6 Conclusions and future work . . . 103

6 Enhancing Underwater Robotics Computer Vision for Autonomous Marine Litter Detection 107
6.1 Introduction . . . 108

6.2 Data preparation and model training . . . 110
6.2.1 Datasets . . . 110
6.2.2 Object detection models . . . 113
6.2.3 Model training . . . 114
6.3 Results & discussion . . . 115
6.3.1 Baseline & generalizability . . . 115
6.3.2 Effect of underwater specific pretraining . . . 117
6.3.3 Evaluating environment generalist model performance . . . 117
6.3.4 Evaluating environment specialist model performance . . . 120
6.4 Conclusions & future work . . . 122

7 Fusing Deep Learning & Physics-Based Models for Classification in Out-of-Distribution Scenarios 125
7.1 Introduction . . . 126
7.2 Related work . . . 127
7.3 Problem statement . . . 127
7.4 Method . . . 128
7.4.1 Appearance-based classifier . . . 128
7.4.2 Model-based classifier . . . 130
7.4.3 Evidence fusion for classification . . . 131
7.5 Case study . . . 131
7.5.1 Ordinary Differential Equation (ODE)-driven dataset . . . 131
7.5.2 Experimental results . . . 132
7.6 Conclusions and future work . . . 135
Appendix . . . 136
7.A System parameters . . . 136

8 Domain-Informed Neural Networks: Detecting Underwater Moving Objects with Noisy Data 137
8.1 Introduction . . . 138
8.2 Related work . . . 139
8.2.1 Datasets and simulators . . . 139
8.3 Proposed framework . . . 140
8.3.1 Mathematical models for incorporating domain-specific knowledge 140
8.3.2 Domain knowledge integration . . . 141
8.4 ODE-driven dataset . . . 143
8.4.1 Generation and curation of data . . . 143
8.4.2 Data preprocessing . . . 147
8.5 Neural network and training . . . 150
8.6 Case studies . . . 152
8.6.1 Supervised learning . . . 153
8.6.2 Semi-supervised learning . . . 157
8.6.3 Computational efficiency . . . 161
8.7 Conclusions and future work . . . 161
Appendix . . . 161
8.A Motion extraction and dataset creation . . . 161

9 Conclusions and Recommendations **163**
9.1 Contributions 163
9.2 Limitations and recommendations for future work 164
9.2.1 Limitations 164
9.2.2 Recommendations for future work 165

Bibliography **169**

Acknowledgements **197**

Curriculum Vitae **199**

List of Publications **201**

SUMMARY

Deep Learning (DL) has transformed the field of computer vision, enabling significant progress in domains such as medical diagnostics, autonomous vehicles, robotics, infrastructure inspection, and industrial automation. However, applying DL-based computer vision techniques in underwater environments remains a considerable challenge. Factors such as light absorption, scattering, and water turbidity degrade image quality, hindering the performance of DL algorithms in such conditions.

This thesis explores the advancement of DL-based computer vision techniques for underwater environments, with a focus on scenarios involving autonomous robotic litter collection from the seabed. The main objective is to address key limitations in this domain, including the scarcity of annotated datasets, poor visibility conditions, and the inherent variability of underwater scenes.

The thesis begins with a review and analysis of state-of-the-art DL methods for video object detection and segmentation, highlighting the potential benefits of integrating domain knowledge to improve robustness in complex underwater settings. To address the issue of data scarcity, an annotated dataset designed for underwater litter detection in shallow waters is presented. This dataset serves as a benchmark for training and evaluating underwater object detection models.

A multi-robot system is then introduced, which integrates aerial, surface, and underwater vehicles to collaboratively detect and collect litter from the seabed. The system architecture, real-world deployment, and the role of computer vision in the operational workflow are discussed.

In addition, an automated image enhancement selection framework is proposed. This framework selects the most suitable enhancement method based on its effect on computer vision tasks such as classification. To improve model generalization across different underwater conditions, environment-specific neural networks are introduced, designed to adapt to differences in turbidity and lighting.

In later chapters, the thesis addresses DL performance in Out-of-Distribution (OOD) scenarios. A hybrid classification system is developed that combines a Convolutional Neural Network (CNN)-based classifier with a physics-based classifier based on the Moving Horizon Estimation (MHE) framework. Dempster-Shafer theory is applied to fuse their predictions, improving classification under uncertainty. Finally, domain-informed neural networks are introduced, where knowledge distillation is used to incorporate physics-based rules into the learning process, improving generalization and reducing the dependence on large-scale labeled datasets.

In summary, this thesis aims to strengthen the DL pipeline for underwater computer vision by improving data availability, integrating DL into robotic systems, optimizing image enhancement, and combining DL with domain knowledge. Each contribution presents an incremental step, with the overall goal to provide robust,

scalable, and practical deployment of DL-based perception systems in underwater environments. In the final chapter, we reflect on how the research objectives and questions were addressed, discuss the limitations of the proposed approaches, and suggest potential directions for future research.

SAMENVATTING

Diepgaande neurale netwerken (in het Engels: Deep Learning, DL) hebben het vakgebied van computervisie getransformeerd en aanzienlijke vooruitgang mogelijk gemaakt in domeinen zoals medische diagnostiek, autonome voertuigen, robotica, infrastructuurinspectie en industriële automatisering. Het toepassen van DL-gebaseerde computervisie in onderwateromgevingen blijft echter een aanzienlijke uitdaging. Factoren zoals lichtabsorptie, verstrooiing en troebelheid van het water verminderen de beeldkwaliteit en belemmeren de prestaties van DL-algoritmen onder deze omstandigheden.

Dit proefschrift onderzoekt de ontwikkeling van DL-gebaseerde computervisietechnieken voor onderwateromgevingen, met bijzondere aandacht voor scenario's waarin autonome robots afval verzamelen van de zeebodem. Het belangrijkste doel is om enkele kernbeperkingen in dit domein aan te pakken, waaronder het gebrek aan geannoteerde datasets, slechte zichtomstandigheden en de grote variabiliteit van onderwaterscènes.

Het werk begint met een overzicht en analyse van de meest geavanceerde DL-methoden voor objectdetectie en segmentatie in video's, waarbij de potentiële voordelen worden benadrukt van het integreren van domeinkennis om de robuustheid in complexe onderwateromstandigheden te verbeteren. Om het probleem van datagebrek aan te pakken, wordt een geannoteerde dataset gepresenteerd die specifiek is ontworpen voor afvaldetectie in ondiep water. Deze dataset dient als benchmark voor het trainen en evalueren van onderwaterdetectiemodellen.

Vervolgens wordt een multi-robotsysteem geïntroduceerd dat lucht-, oppervlakte- en onderwatervoertuigen integreert om gezamenlijk afval op de zeebodem te detecteren en te verzamelen. De systeemarchitectuur, implementatie in de praktijk en de rol van computervisie binnen de operationele workflow worden besproken.

Daarnaast wordt een geautomatiseerd selectieraamwerk voor beeldverbetering voorgesteld. Dit raamwerk kiest de meest geschikte methode op basis van het effect op computervisietaken zoals classificatie. Om de generalisatie van modellen onder verschillende onderwateromstandigheden te verbeteren, worden omgevingsspecifieke neurale netwerken geïntroduceerd die zich aanpassen aan variaties in troebelheid en verlichting.

In latere hoofdstukken behandelt het proefschrift de prestaties van DL in situaties buiten de verdeling (in het Engels: Out-Of-Distribution, OOD). Er wordt een hybride classificatiesysteem ontwikkeld dat een op convolutionele neurale netwerken gebaseerde classifier (in het Engels: Convolutional Nural Network, CNN) combineert met een fysisch gemotiveerde classifier op basis van het raamwerk van bewegende horizoninschatting (in het Engels: Moving Horizon Estimation, MHE). Dempster-Shafer-theorie wordt toegepast om hun voorspellingen samen te voegen,

wat leidt tot een verbeterde classificatie onder onzekerheid. Tot slot worden domeingeïnformeerde neurale netwerken geïntroduceerd, waarbij kennisdistillatie (in het Engels: knowledge distillation) wordt gebruikt om fysisch geïnspireerde regels op te nemen in het leerproces. Dit verbetert de generalisatie en vermindert de afhankelijkheid van grootschalige gelabelde datasets.

Samenvattend beoogt dit proefschrift de DL-pijplijn voor onderwatercomputervisie te versterken door de beschikbaarheid van data te verbeteren, DL te integreren in robotsystemen, beeldverbetering te optimaliseren en DL te combineren met domeinkennis. Elke bijdrage vormt een incrementele stap met als overkoepelend doel het ondersteunen van betrouwbaardere, schaalbare en praktisch toepasbare DL-gebaseerde waarnemingssystemen in onderwateromgevingen. In het laatste hoofdstuk reflecteren we op de mate waarin de onderzoeksdoelen en -vragen zijn beantwoord, bespreken we de beperkingen van de voorgestelde benaderingen, en doen we suggesties voor toekomstig onderzoek.

1

INTRODUCTION

In this chapter, we explore the concept of underwater debris detection and the role of Artificial Intelligence (AI) and robotics in addressing the issue of marine waste. We begin with a short overview of the current state of marine pollution and the limitations of traditional collection methods, emphasizing the need for autonomous solutions. The potential for robotics and AI-based computer vision to manage marine litter is discussed, alongside with the challenges posed by the complex nature of underwater environments, which form the key motivations for this thesis. The research questions that structure this work are also presented. In addition, we provide a summary of the thesis contributions, including the development of novel architectures for computer vision, the creation of a dataset for underwater debris detection, and the implementation of a multi-robot system for autonomous seabed litter collection. The chapter concludes with the thesis structure, describing the focus of each chapter and how they contribute to advancing the field of underwater computer vision.

1

1.1. BACKGROUND

Today's oceans contain 26-66 million tons of waste, of which approximately 94% is located on the seafloor [215]. Efforts for marine waste collection have focused mainly on surface-level waste, while underwater collection has received only little attention. Currently, most initiatives for underwater waste collection rely on human divers. However, this method has limited efficiency, due to the high costs, safety risks, and the physical constraints of human intervention in complex underwater environments.

Marine pollution severely risks marine life, water quality, and human health. Plastics, metals, and other materials are accumulating at an alarming rate, especially in coastal and shallow water regions, where they disrupt marine habitats and release toxic substances into the ecosystem [186] [119]. This environmental crisis is further complicated by the challenges associated with monitoring and collecting underwater debris. Traditional methods, such as manual debris collection, are often inefficient, costly, and labor-intensive, providing only limited coverage.

Recent advancements in autonomous systems and computer vision open new pathways to address these challenges by providing scalable, real-time solutions for underwater debris monitoring and collection. Unmanned Underwater Vehicles (UUVs) equipped with computer vision can minimize human intervention by automating debris detection, localization, and classification. However, adapting Deep Learning (DL)-based computer vision techniques, which achieve almost human-level performance in above-water applications, to underwater scenarios introduces numerous challenges. Underwater environments are highly dynamic, with low visibility, light absorption, and turbidity, all of which affect visual data quality and the performance of DL algorithms [3].

SeaClear — short for SEarch, identificAtion and Collection of marine Litter with Autonomous Robots — is a European Horizon 2020-funded project that aims to address the problem of underwater litter, with the help of autonomous robots and AI. The project employs a mixed team of unmanned underwater, surface, and aerial vehicles to locate and collect litter from the seabed. One of its key objectives is to combine UUVs with DL-based computer vision to detect and remove marine litter, enhancing detection accuracy and achieving a robust, reliable workflow in real-world conditions.

Building on this evolving landscape, this PhD thesis addresses the technical challenges involved in deploying DL-based systems for underwater debris localization and classification, contributing to the development of effective solutions for complex marine environments.

1.2. MOTIVATION AND CHALLENGES

Several key challenges highlight the complexity of applying DL to underwater debris detection and motivate the directions followed in this thesis:

1. **Data Scarcity and Quality**
   High-quality, annotated underwater datasets are limited, especially those

covering diverse marine environments and shallow waters. The lack of standardized datasets limits the ability of DL techniques to generalize in different conditions, such as different turbidity levels, depths, and lighting intensity.

2. **Environmental Variability**
   Underwater environments exhibit significant variability due to factors like lighting, depth, and water movement. This variability changes the visual characteristics of the data, such as color, contrast, and object visibility, which create domain shifts. In other words, the conditions during deployment differ from those in the training environment, leading to degraded performance of DL techniques.

3. **Integration with Autonomous Systems**
   Autonomous underwater vehicles require real-time, efficient algorithms capable of identifying and classifying debris in a manner that supports navigation and collection. Addressing computational constraints with the integration of DL approaches in these systems is a significant challenge.

4. **Coupling of Domain Knowledge with DL Methods**
   DL systems are problematic when it comes to "outliers" that differ substantially from the samples on which they are trained. By coupling domain knowledge with DL architectures, it is possible to ensure reliability, especially in unstructured underwater environments. This integration allows not only to rely on data-driven learning but also to adhere to real-world physical constraints (i.e., movement dynamics). However, integrating data-driven methods with domain knowledge while ensuring physical consistency remains complex, and few existing approaches address this combination comprehensively.

1.3. RESEARCH QUESTIONS

Based on the motivation and challenges, this thesis aims to answer the following research questions, which focus on both theoretical and application-oriented aspects of underwater computer vision:

1. **How can the State-of-the-art (SOTA) techniques in DL for video object detection and segmentation be organized and standardized?**
   Although many works propose novel, sophisticated DL techniques for video object detection, the related research literature is scattered. This research question aims to systematically categorize DL techniques for video object detection and segmentation, analyzing the architectures and the primary challenges inherent in such data-driven methods.

2. **To what extent can dataset expansion and enhancement improve object detection performance across diverse underwater conditions?**
   This question explores data generation and its impact on addressing the limitations of scarce underwater datasets and enhancing generalization among different environmental conditions.

3. **How can autonomous multi-robot systems be effectively deployed for underwater debris detection and collection, and what role do advanced computer vision techniques play in such applications?**
   This question investigates the integration of DL techniques within the SeaClear multi-robot platform, focusing on practical deployment, debris-collection workflow optimization, and the impact of advanced computer vision methods.

4. **How can DL architectures be coupled with dynamic models (i.e., mathematical models describing time-dependent properties, such as object movement) to improve object recognition?**
   This question examines the integration of DL architectures with dynamic models to enhance the robustness of object recognition, especially for handling motion, noisy data, and Out-of-Distribution (OOD) samples.

1.4. CONTRIBUTIONS

This thesis presents contributions across several domains, including underwater debris detection, autonomous robotics, and the integration of DL with domain knowledge for classification robustness. These contributions can be summarized as follows:

1. **Organizing Methods for DL in Object Detection and Segmentation**
   The DL techniques for video object detection and instance segmentation are categorized in a systematic way, focusing on the specific requirements of localization and pixel-level segmentation in videos. Key challenges are identified and it is proposed to address them by combining DL with domain knowledge in hybrid architectures. An additional feature of this contribution is that an analysis and categorization of the currently existing approaches integrating domain knowledge with DL techniques is presented.

2. **Creation and Validation of the Seaclear Marine Debris Dataset**
   This work introduces the Seaclear Marine Debris Dataset, a comprehensive dataset for debris detection in shallow water. This dataset, validated through baseline DL architectures, addresses the scarcity of labeled underwater litter data, providing a crucial resource for advancing research in this field.

3. **Development of an Autonomous Multi-Robot System for Marine Debris Collection**
   The SeaClear system, a multi-robot platform, is presented, integrating DL-based computer vision algorithms with real-time navigation and control methodologies. The system's capabilities are validated through real-world deployment, demonstrating its feasibility for autonomous marine debris collection and highlighting practical insights for evaluating robotic performance in underwater environments.

4. **Evaluation of Image Enhancement Techniques within a Computer Vision Pipeline**

By evaluating image enhancement as part of a complete computer vision pipeline rather than as a standalone component, this work establishes a framework that ensures image quality improvements contribute directly to object detection accuracy. This approach represents a first step for shifting the evaluation criteria of image enhancement techniques towards the entire vision pipeline for underwater applications.

5. **Innovative Approaches for Robust Classification**
   Novel methods to improve DL model generalization are proposed, especially for OOD scenarios, by combining DL with Moving Horizon Estimation (MHE) [2, 220] and by developing domain-informed neural networks trained with knowledge distillation [106]. The Dempster-Shafer (D-S) theory [298] is employed to fuse the input of multiple classifiers, providing a framework for robust methods that maintain accuracy in unstructured and noisy conditions.

1.5. THESIS OUTLINE

This thesis consists of nine chapters, structured into three main parts, as depicted in Figure 1.1. Chapters 2 to 8 contain a collection of papers (published, submitted, or to be submitted). As each paper addresses different aspects of computer vision, the mathematical notations are introduced and defined for each chapter separately.

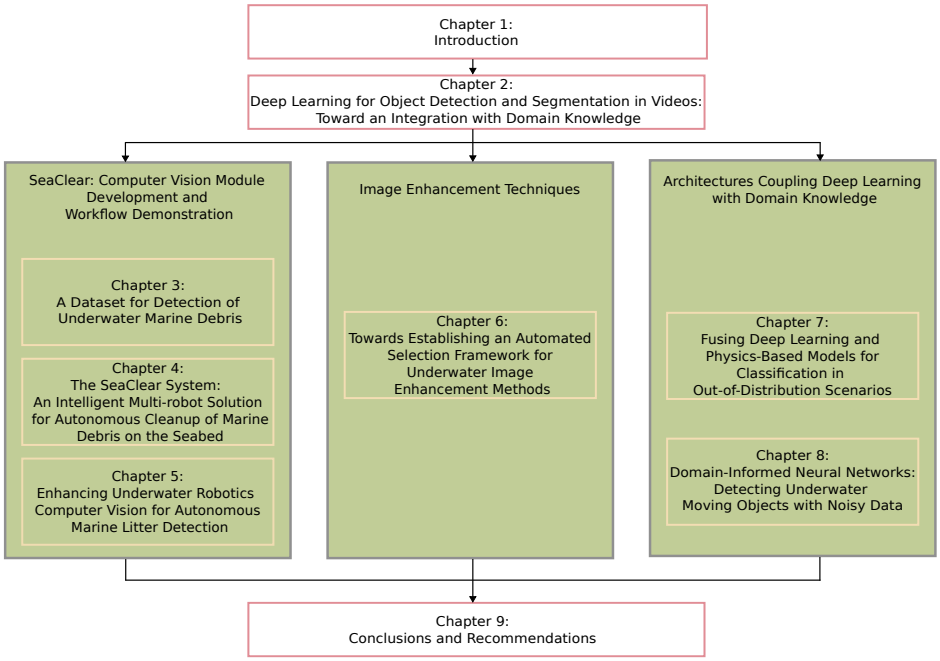


Figure 1.1: Structure of the thesis.

1

This thesis is structured as follows. Chapter 1 provides an introduction to computer vision for underwater environments, discussing the motivation and challenges addressed by this work. Chapter 2 organizes the methods for DL for object detection and segmentation in videos, highlights the current challenges in DL-based techniques, and proposes to integrate DL techniques with domain-specific knowledge to enhance detection performance. Chapter 3 introduces the Seaclear Marine Debris Dataset, analyzing its development, structure, and validation results, as well as the baseline performance of Convolutional Neural Networks (CNNs) on the dataset. Chapter 4 presents the SeaClear multi-robot platform, covering the design, architecture, capabilities, and results from its real-world deployment for autonomous debris detection and collection. Chapter 6 proposes specialist neural networks optimized for particular underwater environments while remaining adaptable to diverse locations of the same environmental type. Chapter 5 proposes a standardized selection framework to evaluate the quality of an underwater image and to estimate the most suitable image enhancement technique based on its impact on the image classification performance. Chapter 7 proposes a fusion of DL classifiers with MHE-based models, using D-S theory to improve classification robustness for OOD scenarios, mainly in noisy data environments. Chapter 8 describes domain-informed neural networks, leveraging knowledge distillation to boost robustness against noisy datasets and enhance computational efficiency. Finally, Chapter 9 concludes the thesis with a summary of contributions, a discussion of limitations, and recommendations for future research directions.

2

DEEP LEARNING FOR OBJECT DETECTION AND SEGMENTATION IN VIDEOS: TOWARD AN INTEGRATION WITH DOMAIN KNOWLEDGE

Deep learning has enabled the rapid expansion of computer vision tasks from image frames to video segments. This chapter focuses on the review of the latest research in the field of computer vision tasks in general and on object localization and identification of their associated pixels in video frames in particular. After performing a systematic analysis of the existing methods, the challenges related to computer vision tasks are presented. In order to address the existing challenges, a hybrid framework is proposed, where deep learning methods are coupled with domain knowledge. An additional feature of this survey is that a review of the currently existing approaches integrating domain knowledge with deep learning techniques is presented. Finally, some conclusions on the implementation of hybrid architectures to perform computer vision tasks are discussed.

This chapter is based on the paper: A. Ilioudi, A. Dabiri, B. J. Wolf, and B. De Schutter. "Deep learning for object detection and segmentation in videos: Toward an integration with domain knowledge". In: *IEEE Access* 10 (2022), pp. 34562–34576. DOI: 10.1109/ACCESS.2022.3162827.
Author contributions statement: A. Ilioudi conceptualized the proposed methodology, reviewed the literature, and wrote the manuscript.

2.1. Introduction

Just as motion perception is essential to our visual system, allowing us to interpret the world, to detect the presence of creatures [75], and to avoid danger [86], video computer vision helps artificial intelligence agents to decipher their surrounding environment and to synthesize actionable information. Inspired by the human visual system and enabled by the latest advancements in Deep Learning (DL), novel video processing methods are emerging that achieve remarkable results and that seek to revolutionize how computer vision tasks are implemented. Yet, similarly to human perception, computer vision is quite prone to illusions.

The fast pace of DL breakthroughs in combination with the improvement in hardware capabilities in terms of computation power, memory capacity, and sensor resolution have accelerated the spread of data-driven methods over the conventional computer vision techniques. Contrary to classical techniques, DL reaches human-level accuracy, requires less expert analysis, and provides superior flexibility including allowing re-training whenever new data are available [282].

The objective of this work is to investigate the advancements of deep learning techniques for computer vision tasks in videos as well as their research perspectives to address their current weaknesses. More specifically, the contributions of our study are trifold:

- We present an analysis of the existing DL techniques for detection and segmentation of objects in videos.
- We present an overview of the challenges with the existing data-driven approaches.
- We outline new directions for research in video processing.

The chapter is organized in seven sections. Section 2.2 presents an overview of necessary preliminary knowledge. Section 2.3 gives a comprehensive overview of DL-based video computer vision methods. In Section 2.4 the current challenges are presented and analyzed. To address these challenges, Section 2.5 presents an overview of approaches that couple DL methods with domain knowledge. Section 2.6 highlights the most prominent topics that are expected to draw major interest from the research community in the following years, and Section 2.7 gives concluding remarks.

A list of abbreviations mentioned in this chapter and their definitions are presented in Table 2.1.

2.2. Preliminaries

In this section, we introduce the most typical tasks of computer vision and we present a brief, comparative analysis between deep learning and conventional techniques in the domain of computer vision, as well as an overview of basic deep learning methods such as convolutional neural networks, restricted Boltzmann machines, and auto-encoders, which constitute the core for DL architectures in computer vision.

Table 2.1: List of abbreviations

Abbreviation	Definition
CNN	Convolutional Neural Network
DFF	Deep Feature Flow
DL	Deep Learning
DM	Dynamics Model
FFT	Fast Fourier Transform
FGFA	Flow-Guided Feature Aggregation
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
IID	Independent and Identically Distributed
LSTM	Long Short-Term Memory
RBM	Restricted Boltzmann Machine
RCNN	Region-based Convolutional Neural Network
RNN	Recurrent Neural Network
SNN	Siamese Neural Network
SSD	Single-Shot Detector
VAE	Variational Auto-encoder
YOLO	You Only Look Once

2.2.1. COMPUTER VISION TASKS

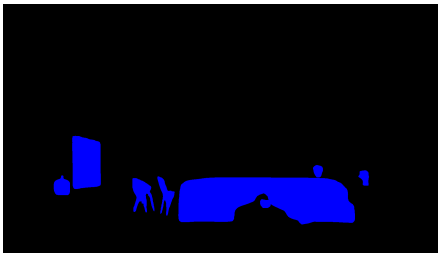
Computer vision tasks can be categorized into 4 major fields: (1) semantic segmentation, (2) classification & localization, (3) object detection, and (4) instance segmentation (Figure 2.1). The task of semantic segmentation refers to the process of assigning a class label to every pixel in an image [172]. One of the shortcomings of this task is the fact that semantic segmentation does not differentiate between instances of the same class. On the other hand, the classification & localization task aims to predict the class of a specific object in an image and to draw a bounding box around the region of the classified object in an image [306]. This task refers to a single object. However, most images in real-world settings contain multiple objects of different shapes and sizes. Therefore, object detection [90] refers to a more general approach where a varying number of predicted objects for every input image can be extracted, since it is unknown how many objects are expected to be detected in each image.

Object detection systems strive to find every instance of an object and estimate the spatial extent of each one. Nevertheless, the detected objects are located just with bounding boxes.

The task of instance segmentation refers to the problem of detecting all the instances of a category in an image and marking the pixels that belong to each one of them [93]. Extending this task to the video domain results in simultaneous detection, segmentation, and tracking of the instances [299]. The instance segmentation task



(a) Classification & localization



(b) Semantic Segmentation



(c) Object Detection



(d) Instance Segmentation

Figure 2.1: Examples of the four major computer vision tasks.

combines object detection, where individual objects are classified and localized with a bounding box, and semantic segmentation, where each pixel is classified into the given classes.

The task of object classification & localization is included in object detection. At the same time, in semantic segmentation, each pixel of an image is associated with a class label like a road, tree, pedestrian, etc. In other words, all objects of an image that belong to the same class are treated as a single entity. On the other hand, each object of the same class is treated as a distinct individual instance with instance segmentation. Hence, instance segmentation can be considered as a more elaborate implementation of semantic segmentation. Since all the computer vision tasks are similar, in this work mainly object detection and instance segmentation techniques will be examined, as they are the most dominant techniques required in extensive applications such as autonomous driving [167], video surveillance [243], face recognition [259], and robot navigation [296].

2.2.2. Deep learning vs. traditional computer vision techniques

Traditional computer vision methods are based on hard-coded, rigid-rule algorithms to apply feature extraction on images [189]. Several algorithms have been developed to extract properties such as corners, edges, and regions of interest from images [13, 29, 94, 174, 205]. These algorithms showcase advantages such as transparency, in terms of allowing to trace back to all steps of how a decision was made, and

performance that is independent of the training dataset. At the same time, however, they have been criticized to be inflexible, difficult to improve or adapt, and highly time-consuming to develop manually for each additional object to be detected [193]. Moreover, the performance of these methods significantly deteriorates when the number of classes to be detected increases. By contrast, DL utilizes massive data sets and numerous training cycles to learn how an object looks, following a process during which relevant features of an object of interest are extracted automatically. The DL architecture can then be implemented on previously unseen images and make accurate predictions. DL-based methods perform remarkably better than traditional methods, albeit with trade-offs regarding computational requirements and training time [193]. As a result, they have vastly replaced traditional computer vision techniques, thanks to their strong ability to be easily adjusted, to extract complex features in much more detail, and to be much more efficient in terms of accuracy and versatility [193]. Tremendous advancements in research have taken place in this domain, resulting in the development of numerous methods. The fundamental DL methods implemented on image computer vision applications are discussed in Section 2.2.3.

2.2.3. IMAGE-BASED DEEP LEARNING METHODS

CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) have been widely used in image processing applications over the past decades [145, 157, 239]. Their structure consists of a number of convolutional and pooling layers, stacked one after another [17]. The convolutional layer can be visualized as a square matrix W of weights, called kernel [204]. The kernel slides over the image looking for patterns and when it distinguishes a part of an image that is similar with its pattern, it returns a large positive value, otherwise, it returns a small value. The input image is represented as a pixel matrix with size length × width × number of color channels (i.e. an RGB image has 3 color channels).

The convolutional layer is utilized for feature extraction and the pooling layer to downsample the resolution of the convolutional layer output. In this way, a dimension reduction is accomplished, which reduces the number of necessary parameters in the next layer, resulting in a less complex architecture. During the training process, the training samples are fed through the CNN and the error with respect to the desired output is calculated. The error and its gradient are then backpropagated through the network layers and the weights are updated.

CNN-based image object detectors can be separated into two main categories [256, 309]:

- **Two-stage approach:** In the two-stage method, the first stage extracts region proposals and the second stage classifies those region proposals and determines the bounding boxes of the classified objects. In the region proposal part, sliding window techniques such as Deformable Part Models [60] are adopted. An additional region proposal technique, employed in Region-based Convolutional Neural Networks (R-CNNs) [77], is selective search

[270]. R-CNNs extract around 2000 region proposals on each input image, which is a significantly reduced number of regions needed compared to other sliding window methods. At the second stage of this architecture, a CNN is used for object detection over the region proposals. The size of the proposed regions is arbitrary, while the CNN requires a fixed-size input. Hence, a major drawback of R-CNNs is due to the fact that images need to be cropped or resized to accomplish the requirement for a fixed-size input. Spatial pyramid pooling [83, 97, 147] is a method used in order to achieve a fixed-size output irrespective of the input image size. Hence, spatial pyramid pooling networks can be trained and tested on varying-size images, which reduces overfitting of the model.

Both R-CNNs and spatial pyramid pooling networks are particularly slow during training. Fast R-CNN [77] tries to solve this drawback by passing the original image through the CNN instead of using the region proposals. As a result, fast R-CNN is faster than R-CNN because the convolutional operation is implemented only once on the original image instead of 2000 times on the region proposals. Fast R-CNNs can train detection networks whose architecture involves multiple layers like VGG-16 [242], as they are 9 times faster compared to R-CNNs and 3 times faster than spatial pyramid pooling networks [256]. The drawback of the high time cost has been further addressed by faster R-CNNs [223]. In faster R-CNNs the time-consuming selective-search algorithm is replaced with a fully convolutional network that learns the region proposals of an image with arbitrary size. A major additional development of the previous R-CNNs is achieved by Mask-R-CNNs [96]. Mask R-CNNs extend the previous architectures by labeling the pixels corresponding to each object instance. The Mask R-CNN inherits the region proposal network from faster R-CNNs and employs an additional branch that outputs a binary mask classifying whether or not a given pixel is part of an object. Two-stage approaches yield a high accuracy since each stage performs one specific task. However, in terms of real-time applications, two-stage approaches show weaknesses in computational time.

- **One-stage approach:** One-stage approaches skip the first stage of region proposal and simply run detection directly on the input image. This simpler architecture allows them to have faster inference. Some networks can achieve a processing speed of up to 150 frames per second (FPS). There is a trade-off, however, in terms of accuracy. Notable one-stage methods are the You Only Look Once (YOLO) network [222], which extracts class and bounding boxes predictions directly from an input image using a CNN and the Single-shot Detector (SSD) [170], which takes an input image and passes it through multiple convolutional layers with different sizes of filters.

Restricted Boltzmann machines

The Restricted Boltzmann Machine (RBM) is a two-layer undirected graphical model [19] that was introduced in 1986 [103]. It consists of a set of visible nodes and a

set of hidden nodes. RBMs are in essence a variant of Boltzmann machines, but in RBMs there are no interlayer connections between the nodes in the visible layer and the hidden layer (i.e. no visible node is connected to any other visible node and no hidden node is connected to any other hidden node respectively). In this way, RBMs are easier to implement and more efficient in training compared to Boltzmann Machines. Their visible nodes receive the input, combine it with weights and a bias, and pass it to the hidden nodes. The value generated at the hidden nodes is combined accordingly with weights and a bias and the result is passed to the visible nodes to reconstruct the input.

If we consider the visible vector V, the hidden vector H, and the weight parameters α_i, b_i, w_{ij}, an RBM configuration can be assigned with an energy E given by [74]:

$$E(V,H) = -\sum_i \alpha_i v_i - \sum_j b_j h_j - \sum_{ij} v_i w_{ij} h_j. \tag{2.1}$$

Given this energy function, a probability P is assigned to every pair (V, H):

$$P(V,H) = \frac{1}{Z} e^{-E(V,H)}, \tag{2.2}$$

where Z is equal to the sum of the energy of all the pairs of visible and hidden vectors.

$$Z = \sum_{(V,H)} e^{-E(V,H)}. \tag{2.3}$$

For a given visible vector V, the probability that is assigned to the hidden node h_j is

$$P(h_j = 1|V) = \sigma\left(b_j + \sum_i v_i w_{ij}\right), \tag{2.4}$$

where $\sigma(\cdot)$ is the logistic sigmoid function [92]. For a hidden vector H the assigned probability of a visible node v_i is respectively:

$$P(v_i = 1|H) = \sigma\left(\alpha_i + \sum_j h_j w_{ij}\right). \tag{2.5}$$

The weight parameters are optimized with the aim to maximize the likelihood of the visible and hidden vectors (V, H).

The intuition behind RBMs is based on the association of a scalar energy to each combination of the variables of interest. Learning is achieved, therefore, by calculating the combination that has the lowest energy.

RBMs are useful for dimensionality reduction, classification, regression, and feature learning. However, due to the fact that RBMs consist of only two layers, the complexity of the data representation that they can achieve is limited [74]. For this reason, a number of extended, architectures has been developed. An example of such architecture is the Deep Belief Network [104], which consists of multiple stacked RBMs. Deep Belief Networks are used for feature extraction in many computer vision

applications. Except for Deep Belief Networks, another RBM-based architecture is the Deep Boltzmann Machine [232, 233]. Deep Boltzmann Machines are similar to Deep Belief Networks, although the former have only undirected connections between their layers, which makes them more robust to noisy observations, while the latter have bidirectional connections in the last layer [254].

Auto-encoders

Auto-encoders [23, 105] refer to a specific type of neural networks that aim to compress the input image data into a lower-dimension (latent) representation and then reconstruct the original image from this representation. Their architecture consists of two main parts, namely, the encoder and the decoder. The encoder maps an input vector of images X into a compressed, lower dimensional vector Z, while the decoder part maps the latent variable Z to a reconstruction of the input image. The encoder and decoder mappings $\phi : \mathcal{X} \to \mathcal{Z}$ and $\psi : \mathcal{Z} \to \mathcal{X}$ are given by:

$$(\phi, \psi) = \underset{(\phi,\psi)}{\operatorname{argmin}} \ \|X - (\psi \circ \phi)(X)\|^2, \tag{2.6}$$

where the operator $\circ$ refers to the composition function: $\psi \circ \phi(X) = \psi(\phi(X))$. The autoencoder is trained with the objective to select the optimal encoder and decoder functions so that the minimum amount of information is required to encode the image in order to be regenerated on the decoder side.

The compressed representation captures the most informative features of the input data in a lower-dimensional space. This latent space can then be used in a variety of computer vision tasks. For example, in classification, the encoder ϕ can serve as a feature extractor so that a classifier is trained on the lower-dimensional $Z = \phi(X)$. This approach is advantageous when labeled data are limited, as the learned features often generalize better and contain less noise. In addition, the latent representation is also useful for anomaly detection, where an input that reconstructs poorly is flagged as a potential anomaly.

2.3. Deep learning methods for object detection and segmentation in videos

Due to the similarity between video detection and image detection, some methods of image detection are often used for video detection. The methods described above can be extended to the video domain by running detection for each image in a sequence of frames [20]. In this way, however, the temporal correlation between frames is not taken into account. In addition, running a detection algorithm for each frame results in computational inefficiency since there might be feature extraction redundancies between sequential frames. Furthermore, in a video sequence, there might be poor-quality frames which could lead to low inference accuracy. One obvious reason why extending methods for image detection to video detection is not trivial is due to the fact that a video sequence introduces an additional dimension; the temporal one. In other words, instead of being considered as a sequence of frames, a video should be rather regarded as a sequence of related frames.

Due to the complexity of video data and the computation cost for training, research has been limited in this field. However, more and more video-related research works have surfaced lately, due to the release of ImageNet VID [229] and other massive video datasets. Depending on the architecture, DL-based techniques for video object detection can be broadly diversified into six categories, namely (1) optical flow, (2) tracking, (3) long short-term memory, (4) gated recurrent unit, (5) self-attention mechanism, and (6) generative learning. In the following subsections, a critical appraisal of these architecture paradigms is presented.

2.3.1. OPTICAL FLOW

One of the most fundamental concepts in video processing is optical flow. Optical flow was originally introduced in [75] referring to human perception and the changing pattern of light that reaches our eyes. In computer vision applications, optical flow refers to the problem of estimating the displacement vector for each pixel in subsequent image frames [111].

A key assumption in optical flow is brightness constancy. This practically means that a pixel at the position (x, y) of an image at time t moves to the position $(x+\Delta x, y+\Delta y)$ at time $t+\Delta t$ and the brightness $I(x, y, t)$ remains constant:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) \approx I(x, y, t). \tag{2.7}$$

The Taylor series expansion of the left-hand side of (2.7) is

$$\begin{aligned} &I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \cdots \Rightarrow \\ &I(x+\Delta x, y+\Delta y, t+\Delta t) - I(x, y, t) = I_x \Delta x + I_y \Delta y + I_t \Delta t, \end{aligned} \tag{2.8}$$

where I_x, I_y, I_t are the partial derivatives of the intensity function I with respect to x, y, and t respectively. Hence, if we substitute (2.7) into (2.8) we can derive:

$$\nabla \mathbf{I} \cdot \mathbf{v}^T + I_t = 0, \tag{2.9}$$

where $\nabla \mathbf{I} = (I_x, I_y)$ and $\mathbf{v} = (\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})$ are the components of the optical flow, and I_t is the temporal gradient of the intensity function.

Optical flow can be applied to estimate the motion of detected objects in video segments by assigning an optical flow vector to the pixels corresponding to the detected object.

Optical flow can be either "sparse" or "dense". Sparse optical flow estimates the flow vectors of some specific features, such as corners or edges of an object within an image frame. Dense optical flow, on the other hand, includes the flow vectors of all the pixels in an image frame. The latter method achieves higher accuracy than the former, although at the cost of increased computational requirements.

Recently, modern CNN architectures have been successfully used for optical flow estimation applications [49]. CNNs can be trained to run on pairs of images and to predict the optical flow field. These flow networks are employed in computer vision tasks for videos according to two different approaches. In the first approach,



(a) Frame 1



(b) Frame 2



(c) Optical flow map

Figure 2.2: Illustration of optical flow computed between two consecutive frames.

one neural network is responsible for the task of object detection and it is applied on sparse key frames. The extracted feature maps from these key frames are then propagated to the next frames with a flow network. This technique is called Deep Feature Flow (DFF) [314] and it achieves great computational efficiency due to the fact that it implements the object detection task only on key frames.

The second approach involving flow networks is known as flow-guided feature aggregation (FGFA) [313]. In FGFA, a feature extraction network is run on all individual frames to create the respective feature maps per frame. The inference at a reference frame is enhanced with an optical flow network that predicts the motion between the neighbor frames and the adjacent frames. The propagated feature maps from neighbor frames are aggregated with the feature map from the reference frame in an adaptive weighting method. FGFA achieves higher inference accuracy but at a higher computation time compared to DFF. For this reason, an impression network [102] is another proposed architecture that combines the two abovementioned techniques, with the objective to take advantage of both methods. Sparse key frame feature maps are then aggregated with other key frames feature maps and at the same time, they are propagated to other non-key frames. The impression network overcomes DFF both in terms of accuracy and inference speed. It is also faster than FGFA although it achieves a slightly lower accuracy level. An alternative architecture, which outperforms FGFA, is proposed in [47], where a two-stream feature aggregation approach is integrated into a one-stage detector to achieve video object detection. In particular, the first stream applies optical flow to estimate the motion and to aggregate the features along the motion path, while the second stream predicts the features of the frame of interest by spatio-temporal sampling and aggregation of features from the adjacent frames. The final predictions result from blending the outcomes from the two streams.

2.3.2. TRACKING

Visual tracking can be described as the problem of estimating an unknown target trajectory over a sequence of image frames [184]. Traditional methods employ a variety of tracking algorithms, such as mean shift algorithm [35], particle filtering [245], and Kalman filtering [132]. With the advancements in data science in recent years, novel DL-based visual trackers have been developed.

Object tracking outperforms optical flow in accuracy [311]. This can be explained by the fact that tracking uses shared networks to achieve feature extraction for detection and tracking. Hence, the requirements in terms of computational power are limited and at the same time, the fusion between the two tasks is performed in a more straightforward way, which achieves higher accuracy compared to optical flow-based models.

CNN is the first architecture that was adopted for DL-based visual tracking. In [59], a region-based fully convolutional neural network [42] is used for jointly performing detection and tracking in an integrated framework. The model is fed with a set of two consecutive image frames, from which the convolutional feature maps are computed. Object detection is run on each frame and a regressor is employed to compute the box transformation from one frame to the other. CNN-based object

tracking models showcase some weaknesses in performance though, due to the scarcity of labeled data in terms of including sets of two consecutive frames, which are necessary for their training, as well as their speed limitations with respect to real-time applications [188].

A baseline approach presented in [299] extends the Mask R-CNN to include an additional tracking branch with an external memory for tracking object instances across frames. The proposed architecture extracts the classification, the bounding boxes, and the segmentation predictions of Mask R-CNN, and it takes into account the past frame information only for tracking. In this way, the task of instance segmentation is extended to videos. CrossVIS [302] presents a novel, cross-frame learning approach that uses the features of an instance in the current frame to segment the same instance in other frames. Crossover learning is integrated with the instance segmentation loss as an objective to obtain cross-frame instance segmentation consistency, achieving a low computational cost. CrossVIS outperforms MaskTrack R-CNN [299] in terms of both accuracy and speed [302].

An additional DL-based method for tracking arbitrary objects involves Siamese Neural Networkss (SNNs) [261]. SNNs have been extensively implemented on visual tracking applications in the past years [15]. An SNN is basically a two-stream network that takes as input pairs of the target and search image and outputs a similarity map. In other words, SNNs learn a function $f : (z, x) \rightarrow f(z, x)$ which compares an image z with a candidate image x returning a high score when the two images are similar with each other. The position tracking of an object can thus be determined by checking all possible locations and selecting the one that corresponds to an image with the highest similarity to the previous frame. SNNs can learn the function f from a training video dataset with labeled object trajectories and they are one of the most promising methods for object tracking due to their performance and efficiency.

Recurrent Neural Networks (RNNs) [79] are an alternative architecture employed in visual object tracking applications. RNNs can be considered to operate on a sequence that contains vectors $x(t)$ and each vector can describe e.g. an image frame from a video at time step t. In other words, an RNN is a neural network that is specialized for processing a sequence of values $x(1), \ldots, x(n)$, where n is the length of the sequence, in a similar way as a convolutional network is specialized for processing a tensor representing an image. The same update rule is applied to each part of the output, resulting in the sharing of parameters through a deep computational graph. RNN-based methods can be considered as a suitable method for visual object tracking since they take into account both spatial and temporal features of video frames [303]. The RNN-based methods aim to improve the tracking performance by utilizing temporal information such as past states of the target's position. However, their implementation is limited because their complex architecture involves a significant number of parameters that need to be determined [164].

2.3.3. LONG SHORT-TERM MEMORY

Although RNNs are naturally suited to time series data, like videos, their implementation suffers from various weaknesses. First of all, while they take into consideration information from the previous time stamp, their performance is deteriorated, when storing information for a longer time period [142]. Sometimes, certain information stored at long past time step might be required to accurately predict the current output. RNNs in that cases are incapable of utilizing such "long-term" dependencies. In addition, RNNs do not have the possibility to keep part of the past time stamp information and to discard the rest. An additional challenge in RNNs is that gradients propagated through the network tend to either vanish or explode because of the repetition of the weight matrix over all recurrent units. At the same time, optical flow techniques make use of temporal information only on two adjacent frames without using temporal information from other previous frames. Long Short-Term Memory (LSTM) [107] is an improved type of RNN that is capable of utilizing long-term dependencies.

The architecture of an LSTM cell is depicted in Figure 2.3. LSTMs are cells consisting of three parts which are known as gates. The first gate determines what part of the information coming from past time steps needs to be "remembered" or can be "forgotten". The second gate inputs information of the current time step to the cell. Finally, the third gate passes the updated information from the current time step to the next one. The first gate is called forget gate while the second and the third ones are called input and output gates respectively.

In the following equations $f(t)$, $i(t)$, $o(t)$ represent the forget, input and output gate vectors respectively, σ is the sigmoid function, $W^{(j)}$ and $b^{(j)}$ refer to the weights and biases corresponding to the j-th gate's neurons, $h(t-1)$ refers to the output of the previous cell at time stamp $t-1$, and $x(t)$ represents the input at time t [113].

- Forget gate

$$f(t) = \sigma\left(W^{(f)}[h(t-1), x(t)] + b^{(f)}\right) \tag{2.10}$$

- Input gate

$$i(t) = \sigma\left(W^{(i)}[h(t-1), x(t)] + b^{(i)}\right) \tag{2.11}$$

- Output gate

$$o(t) = \sigma\left(W^{(o)}[h(t-1), x(t)] + b^{(o)}\right) \tag{2.12}$$

Moreover, an additional vector $\bar{C}$ is used that modifies the cell's state C:

$$\bar{C}(t) = \tanh\left(W^{(c)}[h(t-1), x(t)] + b^{(c)}\right) \tag{2.13}$$

$$C(t) = f(t) \odot C(t-1) + i(t) \odot \bar{C}(t), \tag{2.14}$$

where the operator $\odot$ corresponds to the elementwise multiplication. The hidden state is equal to:

$$h(t) = o(t) \odot \tanh C(t). \tag{2.15}$$

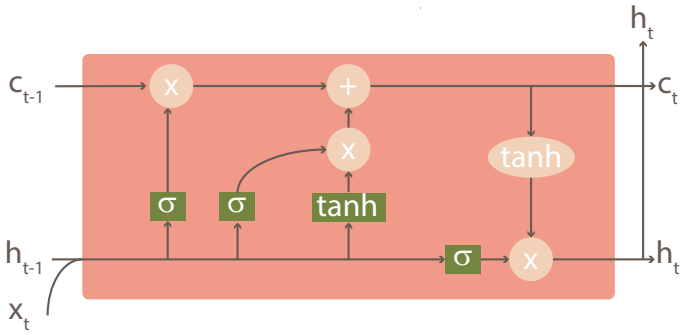


Figure 2.3: LSTM cell structure, adapted from [305].

LSTMs can maintain important information over a long sequence of data. [85] presents an extensive analysis of variants of LSTM as well as a review of the impact of the involved hyperparameters. In [176] an LSTM framework is developed as an extension to an SSD architecture in order to associate detected object instances across consecutive frames. The proposed method outperforms other RNN architectures [268] and it can be applied online. However, the weakness of this approach is that the SSD architecture involved is pre-trained in advance and thus, the SSD features do not get updated in response to the output of the LSTMs. In [169], an approach is suggested where LSTM is used in combination with interleaving conventional feature extractors with extremely lightweight ones. The main advantage of this approach is that minimal computation is required to produce accurate detection. In other words, an interleaved model framework is proposed, where multiple feature extractors are run sequentially or concurrently. A memory mechanism is then proposed to aggregate these frame-level features. A modified LSTM cell is used in [312] to achieve faster results with low computational requirements. The proposed architecture connects fast single-image object detection frameworks in series with convolutional LSTM layers in order to propagate frame-level information over time. This architecture inputs one single frame of the video at a time and it is quite simple. Hence, it achieves reduced computational cost as well as enhanced inference speed.

2.3.4. Gated Recurrent Unit

Similarly to LSTMs, Gated Recurrent Units (GRUs) [33] are another type of RNNs. However, GRUs have fewer parameters than LSTMs, since they only have two gates: the update gate and the reset gate. As seen in Figure 2.4, in contrast to LSTMs, a GRU cell does not have an output gate, as in the case of LSTMs, and they combine

the input and the forget gate of LSTMs into the update gate. Due to their simplicity, GRUs are significantly faster rather than LSTMs.

The update and reset gates in a GRU cell are defined as in equations (2.16) and (2.17) respectively. In the following equations, $z(t)$, $r(t)$ represent the update and reset gate vectors respectively, and $W^{(j)}$, $b^{(j)}$ refer to the weights and biases corresponding to the j-th gate's neurons [113].

- Update gate

$$z(t) = \sigma\left(W^{(z)}[h(t-1), x(t)] + b^{(z)}\right) \tag{2.16}$$

- Reset gate

$$r(t) = \sigma\left(W^{(r)}[h(t-1), x(t)] + b^{(r)}\right) \tag{2.17}$$

The update gate determines the amount of previous time-step information that passes along the next state, while the reset gate is responsible for deciding what part of the past information is neglected. After multiplying the input vector and the hidden state with the weights of the reset gate as presented in (2.17), the element-wise product between the reset gate and the previous time-step hidden state is calculated. Then, a non-linear activation function is applied to the result leading to the candidate hidden state:

$$\bar{h}(t) = \tanh\left(W^{h}[r(t) \odot h(t-1), x(t)] + b^{(h)}\right). \tag{2.18}$$

The hidden state then reads as:

$$h(t) = \left(1 - z(t)\right) \odot h(t-1) + z(t) \odot \bar{h}(t). \tag{2.19}$$

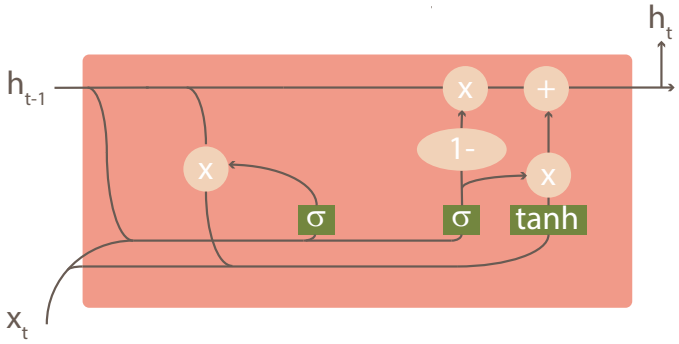


Figure 2.4: GRU structure, adapted from [305]

In [25] an SSD-based architecture is extended to multi-frame data. Convolutional GRUs are employed in order to fuse features across multiple frames and to enhance the accuracy of object detection. From a mathematical perspective, this architecture

replaces the dot product operator in the standard gated recurrent unit definition in (2.16)-(2.18) with the convolution operator. As reported in [72], this approach improves the existing SSD architecture by 2.7 % in terms of the mean average precision on the KITTI dataset [73]. An additional example is provided in [268], where first a pseudo-labeler is trained on individual labeled frames. The pseudo-labeler assigns the labels to all video frames and then a recurrent architecture with GRUs is trained, which takes sequences of pseudo-labeled frames as input. The standard cost function used for the training of the RNN is augmented with an additional term to ensure the consistency across consecutive frames. In [271] a human activity recognition technique is proposed, where skip connections are introduced among GRU layers to ensure that even in a deep architecture with multiple layers, there is no vanishing gradient impact on the performance.

Both LSTM and GRU can ensure that important information is maintained along long time-series data. GRU is faster than LSTM in terms of training speed [301]. Their performance is comparable, although in small datasets, GRU slightly outperforms LSTM.

2.3.5. Self-attention mechanism

RNNs, LSTMs, and GRUs have been widely adopted in sequence modeling applications. However, due to the fact that they process the data in a sequential manner, they do not allow for parallel computation, which could critically affect long sequences of frames, due to memory constraints limiting the batch size of samples during training.

Self-attention mechanism [138] relates different elements of a sequence to generate a representation of this sequence. Contrary to the architectures mentioned above, it supports parallel processing of sequential data. Originally it was proposed for machine translation [278] and then its application was extended to video data [76].

Three vectors are involved in the self-attention mechanism. These vectors are used for the representation of features (key vector), values (value vector), and the values to be determined (query vector). Let us assume that we have a sequence of n elements $(x_1, x_2, \ldots, x_n)$ of $\mathbf{X} \in \mathbb{R}^{n \times d}$, with d being the embedding dimension for the representation of each element [136]. We can then define three learnable weight matrices in order to transform the queries ($\mathbf{W}^q \in \mathbb{R}^{n \times d_q}$), keys ($\mathbf{W}^k \in \mathbb{R}^{n \times d_k}$) and values vectors ($\mathbf{W}^v \in \mathbb{R}^{n \times d_v}$). In this way, the input $\mathbf{X}$ is first transformed with the weight matrices and projected onto $\mathbf{Q} = \mathbf{X}\mathbf{W}^q$, $\mathbf{K} = \mathbf{X}\mathbf{W}^k$, and $\mathbf{V} = \mathbf{X}\mathbf{W}^v$. A similarity function is used to calculate the similarity between the query and the key vector. The self-attention layer outputs $\mathbf{Z} \in \mathbb{R}^{n \times d_v}$ which is equal to

$$\mathbf{Z} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_q}}\right)\mathbf{V}, \tag{2.20}$$

where the softmax function is defined by

$$\text{softmax}(\mathbf{X})_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}, \tag{2.21}$$

for $i = 1 \dots k$ and $\mathbf{X} = (x_1, \dots, x_k) \in \mathbb{R}^k$. The self-attention determines the similarity between the key and the query vector by computing their dot product. The dot product is then normalized using softmax so that the sum of all the scores becomes equal to 1. Each element is then given by the weighted sum of all elements in the sequence. The weights in this case correspond to the attention scores. The most well-known, self-attention architecture is the transformer [278].

In [76] a transformer framework is developed to recognize and localize human actions in a video. A person feature is represented as the query (Q) and the features from adjacent video frames correspond to the key (K) and the values (V). A video instance segmentation architecture built upon transformers is proposed in [284]. Four modules are included in the developed architecture: a backbone CNN to extract features over the video frames, an encoder-decoder transformer that determines the similarity of features on pixel and instance level, an instance-sequence matching, and a segmentation module. The overall performance of this framework is competitive compared to the single-model approaches tested on the YouTube-VIS dataset [299], although it is somewhat lower in comparison to other complex CNN-based models [14].

In [87] a constrained self-attention architecture is proposed for video object detection that captures motion cues under the assumption that moving objects follow a continuous trajectory. An additional, self-attention based architecture is proposed in [89], which is applied in the temporal-spatial domain towards aligning two feature maps of consecutive frames. The proposed method features a low amount of parameters, while it achieves higher accuracy in comparison to optical flow-based methods such as DFF and FGFA. A related, efficient, and simplified architecture for video object detection via aggregating semantic features across frames is presented in [293]. Cosine similarity is implemented to compute the semantic similarities of the extracted proposals across frames, which are then aggregated accordingly. In [46] an object relation module is employed as part of a multi-stage architecture, in order to extract object relations in both spatial and temporal context. The relations are then further distilled with refined supportive object proposals and propagated across frames. Finally, in [241] an attention-based module is developed to learn long-range temporal relations between objects, in order to propagate the extracted features. The proposed architectures in [46, 293], and [241] outperform optical flow-based approaches in accuracy.

2.3.6. GENERATIVE LEARNING

The objective of generative learning is to approximate a complex, high-dimensional probabilistic distribution that generates a class of data, in order to generate similar data. Developing generative architectures to understand complicated data distributions has been a long-standing research problem [196]. Recent works in this area [80, 140] have provided a new set of generative algorithms that can efficiently generate video segments or extract features from them. The most outstanding generative algorithms are the Variational Autoencoders (VAEs) and Generative Adversarial Networkss (GANs).

- **Variational auto-encoders:** Their architecture resembles an auto-encoder, with the difference that their latent variable distribution is regularized during the training. VAEs stemmed from the limitation of auto-encoders to generate new, unseen data, due to the fact that the distribution of the latent variable is unknown. To alleviate this issue, VAEs are trained to learn the distribution of the latent variable, assuming that it follows a Gaussian distribution with a mean μ and variance σ^2 [121]. One example of a VAE-based architecture for video object detection is presented in[161], where a modified VAE architecture, built on top of a Mask R-CNN is proposed, in order to detect and to segment multiple instances in diverse videos. The proposed architecture outperforms MaskTrack R-CNN [299], because the MaskTrack R-CNN architecture depends entirely on the Mask R-CNN to perform predictions, resulting in difficulties to handle false negative proposals of the Mask R-CNN in highly diverse videos with occlusions, deformations, and pose variations of objects. By contrast, the architecture proposed in [161] merges a VAE with a Mask R-CNN network in a topology consisting of one encoder and three decoders. This results in three parallel branches that provide strong complements for predictions about bounding boxes and mask features, and they significantly reduce the number of false negatives in the Mask R-CNN module.

- **Generative adversarial networks:** Generative adversarial networks are built on the basis of a two-player, min-max game. The generator network G and the discriminator network D correspond to the first and the second player respectively. The generator's objective is to mislead the discriminator by generating natural-looking data (e.g. images, videos, etc.) from a random, latent vector z. The discriminator on the other hand, tries to distinguish whether the data are real or fake (generated). The game is modeled as the following optimization problem:

$$\min_G \max_D (G,D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]. \tag{2.22}$$

  A generative adversarial approach is developed in [249], to randomly generate masks that correspond to object appearance variations in time. The masks are then applied to reduce overfitting via adaptively dropping out input features. The developed architecture identifies the mask that maintains the most robust features of the target objects over a long period of time. In [257] a GAN is trained on color and depth information in order to generate similar backgrounds to the test samples. The generated background samples are then subtracted from the given test samples to detect foreground moving objects. Finally, in [28] the encoder-decoder architecture of [194], which is limited to process information between only two adjacent frames, is extended with a GAN, to enforce temporal and spatial coherence of the generated object masks and to exploit information within a longer temporal window. The developed architecture exhibits similar accuracy as other state-of-the-art computer vision methods, while it is almost four times faster.

2.4. CHALLENGES IN DEEP-LEARNING-BASED COMPUTER VISION

Despite the tremendous advances in deep learning and the fast pace of its breakthroughs over the last years, there are still challenges that prevent it from reaching its full potential. This section illustrates a set of major challenges related to computer vision tasks on video analysis with DL techniques.

DL-based methods have succeeded in achieving even human-level performance in complex, computer vision tasks. However, this is possible only when massive datasets are available for training. Data are the core of any DL-based process and hence their shortage is often responsible for poor performance. Large-scale amounts of data are not available for all video applications though.

The impact of data scarcity is further escalated by the stand-alone approach of DL. A typical workflow for developing a DL module consists of creating a training set of inputs associated with outputs and learning the relations between them. In this way, however, the architecture becomes free-standing and isolated from prior, useful knowledge. Hence, the DL performance is highly determined by the existence of big-volume datasets while at the same time, applications that are more related to common sense reasoning and less to categorization, cannot be sufficiently targeted with purely DL methods [180].

Generalizability is an additional major challenge concerning the performance of a data-driven model trained on one dataset when applied to other datasets. When training deep neural networks with high complexity and numerous parameters, the cost function might have multiple minima, which minimize the training error but may not generalize well to unseen data. The presence of noise and outliers in the training dataset is an additional reason for poor generalizability. Generalizability is also deteriorated due to the weakness of DL methods to deal with hierarchical structures, since DL modules tend to fail when generalization depends on compositional processes [146].

At the same time, although correlation does not imply causation, they do not seem to be distinguishable for DL. Numerous neural network architectures have surfaced over the last decades that are highly capable of discovering complex correlations in data, yet they lack in reasoning about cause-effect relations or environmental changes.

Finally, deep learning has delivered new, highly performing approaches in computer vision tasks, whose dominance, however, remains inversely proportional to their explanatory power. Rationalizing the output of data-driven techniques is a critical issue since more and more data-driven systems are adopted in safety-critical and high-impact applications.

2.5. Integrating deep learning with domain knowledge

2.5.1. Motivation

A prudent approach to address the abovementioned challenges is to expand the current methods and to merge them with principles that govern the dynamic behavior of systems over the time, enabling an adaptation to new, unseen scenarios. Combining DL-based techniques with equation-based dynamic models (DMs) in a complementary way, or in other words, integrating common sense understanding into artificial intelligence constitutes a particularly interesting challenge for computer vision systems.

Enabling data-driven vision systems to understand the principles that govern the behavior of objects is essential for the development of autonomous systems that understand observed scenarios and have the ability to adopt these principles to a never seen situation. Leveraging domain knowledge to identify equation-based models that describe how the properties of objects and entities change over time and embedding them into DL techniques can lead to novel, highly robust, and performing architectures. Such models could be developed for instance from well-known first principles in order to describe how an object moves and they could be coupled with DL methods forming a hybrid computer vision architecture. It is straightforward to conclude that hybrid architectures are more efficient compared to purely data-driven or model-based techniques as they harness the benefits of both disciplines. Hybrid methods that combine scientific domain knowledge with data-driven models allow for accurate inference even with imperfect models and limited amounts of data.

The integration of the two disciplines in a hybrid architecture can be realized either by infusing mathematical rules to a DL architecture or by combining the operation of the two separate modules in a complementary manner. An advantage of this second version of a hybrid architecture is the fact that an easy and straightforward recalibration of the DM module is feasible if a bidirectional interaction between the two modules is enabled. More specifically, the DL module, which can be re-trained incrementally when new data become available, can also enable the recalibration of the DM module. This results in a hybrid architecture which is highly flexible and easily adaptable to different scenarios.

Hybrid architectures merging data-driven techniques with domain knowledge, such as from physics have been recently developed, introducing a novel research field which is still in its infancy [133, 216]. As a result, their applications are limited mainly to topics related to climate science and geology. Their expansion to other disciplines like computer vision tasks remains a challenging research topic but would undoubtedly contribute towards addressing the abovementioned impediments in purely data-driven methods.

2.5.2. Hybrid architectures

A taxonomy of four general classes for integrated data-driven and model-based techniques can be derived. This classification is based on the level at which the

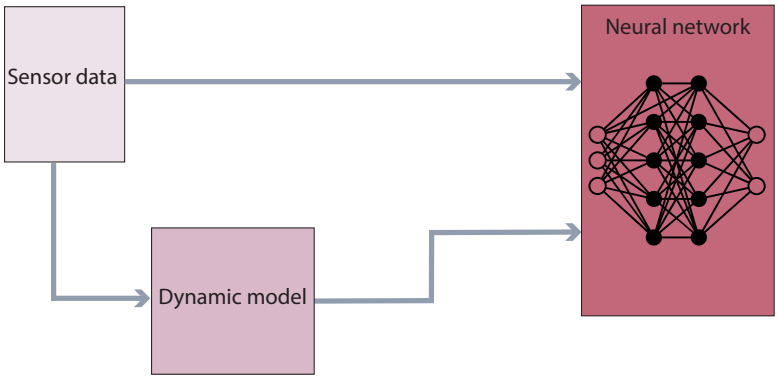


Figure 2.5: Hybrid architecture. The dynamic model could refer to a first-principles model or any other mathematical or computer model that is derived from domain knowledge and that describes how the properties of objects and entities change over time.

integration takes place [133], [216]. More specifically, the four classes are: (1) preprocessing level, (2) initialization, (3) design of architecture, and (4) regularization. This section presents an analysis of these different methodologies.

DATA PREPROCESSING LEVEL

Data preprocessing is essential in all data-driven techniques before passing the data through the DL module. The reason is straightforward: the quality of data determines the information that can be extracted and hence, it directly influences the learning process of the DL algorithm. As a result, it is vital that we apply a preprocessing technique before passing the data through the DL model.

The concept of data preprocessing is a major area in the field of deep learning. There are three main steps involved in data preprocessing, that is: (1) data cleaning, (2) data transformation, and (3) data reduction. Data cleaning refers to the handling of missing data as well as, to noise removal. Data transformation may include normalization of the data, band-pass filtering, downsampling, and feature selection. When the input involves time-series signals, the data can be converted to the frequency domain via the fast Fourier transform (FFT). This implementation can be applied in anomaly detection such as e.g. in the bearings of a rotating machine [231]. Finally, reducing the dimension of the feature set is another technique widely applied when preprocessing the data. A thorough analysis of the data preprocessing techniques is presented in [71].

INITIALIZATION

One important design choice when building a neural network architecture is related to the parameter initialization [290]. Iterative optimization algorithms such as gradient descent are used during the process of training a neural network in order

to estimate the network's parameters. In this process, an initial value for the parameters is required as a first step to start the optimization process. Quite often the initialization of the parameters is done based on a random distribution. Random initialization though can make the optimization algorithm that is employed for the calculation of the network weights to converge to local minima or saddle points.

An approach towards this issue would be to use a technique called transfer learning [198]. The basic idea of transfer learning is based on pretraining a neural network on a simpler, related problem. This pretraining task takes place under the assumption that a big quantity of data is available. This pre-trained neural network can then be implemented as the initial state for the training of the original problem as it is closer to the optimal parameters value than random initialization. Transfer learning is a widely used technique in complex DL applications such as natural language processing and computer vision. However, the performance of this technique is highly dependent on the availability of big-scale data. An alternative approach is to employ domain-specific knowledge to assist the selection of the initial values of the parameters involved [133]. In this way, first-principles models can be used to generate approximate simulations for the initialization of the parameters of the neural network. Domain knowledge can ensure a reliable initialization of the parameters, which can assist in achieving generalizable, interpretable, and physically consistent architectures.

Design of architecture

Data-driven techniques have made a major impact at realizing highly performing systems for solving hard problems related to pattern recognition, prediction, etc. However, a major impediment in their wide adoption in critical applications is their "black box" nature since our understanding of their complexity is limited. Hence, domain knowledge can be infused in a DL architecture to ensure its interpretability.

One possible approach for this integration is to infuse the output of the equation-based model f_{DM} as input to the DL module f_{DL}, i.e. $f_{\text{hybrid}} : (X, P_{\text{DM}}, P_{\text{DL}}) \rightarrow Y$ where X is the input, Y is the output, P_{DM}, P_{DL} the parameters of the dynamic model and the DL model respectively, and f_{hybrid} the composition of the two functions, $f_{\text{hybrid}} = f_{\text{DL}} \circ f_{\text{DM}}$ [216].

Two main categories of architectures can result from merging DL with dynamic models, founded on prior domain knowledge. In the first category, the output of the model is fed through the DL module at the first or at an additional layer. In the second category, the model is embedded into the DL module. Many architectures with respect to the first class have surfaced lately in the field of climate and geology applications. In [128, 134], the output of a physics-based model is provided as an additional input feature to the DL module in an application related to predicting the temperature of a lake based on the depth. In [202], a physics-based neural network architecture is used in order to simulate broadband earthquake ground motions. The DL module is used to predict the ground motion in the short term, including transient effects, which are particularly complex to model mathematically. The DM module is then used to simulate the response in a long-term period.

In the second class, the DM module is embedded into the DL module architecture.

An example of this class is a physics-based model with an RNN including LSTMs [244] where the sensor data as well as the DM generated output are ingested as input to the RNN architecture.

REGULARIZATION

Deep neural networks can involve numerous parameters. However, when no large amounts of data are available, deep neural networks tend to overfit or, in other words, they fail to discover the underlying relationship described by the training data and hence they cannot extrapolate to observed data outside the training set. One way to handle this issue is to apply physical constraints on the loss function of the neural network. Several regularization techniques have been developed in this way, to prevent neural networks from overfitting. This is achieved by applying penalties to layer parameters, and by integrating these penalties in the loss function that is minimized during training. The loss function in that case will be of the following form [290]:

$$f_{\text{Loss}} = f_{\text{Trn}}(Y, \hat{Y}) + \lambda R(W) + \gamma f_{\text{Phy}}(\hat{Y}), \tag{2.23}$$

where f_{Trn} corresponds to a function that represents the error between the predicted value $\hat{Y}$ and the true value Y. This function can be for example the mean squared error or cross entropy. In addition, λ represents a hyperparameter determining the weight of the regularization term $R(W)$. The first two terms of (2.23) describe the standard loss function used when training a neural network. The additional term f_{Phy} corresponds to the physics-based constraint and it aims to ensure the consistency of the trained system with first-principles laws or dynamic models. The weight of this function is represented by the hyperparameter γ. Given the true value Y, the following is considered as the general optimization problem to solve for (2.23):

$$\underset{w}{\text{argmin}} \quad f_{\text{Trn}}(Y, \hat{Y}) + \lambda R(W) + \gamma f_{\text{Phy}}(\hat{Y}). \tag{2.24}$$

By introducing model-based constraints in the loss function for the training of DL modules, scientific consistency is achieved, which is essential for training generalizable models. In addition, the physics-based loss function f_{Phy} requires no labeled data which allows the training of the DL module to be expanded to non-labeled data. A plethora of implementations that impose physics-based constraints on the training of DL modes has surfaced recently [190, 252, 258]. In [134] a physics-based loss function is used for the training of a temperature lake predictor. The loss function encompasses a constraint resulting from the relationship between the temperature, the density, and the depth of the lake water. In this way, the trained predictor achieves enhanced generalizability while at the same time, consistency with first-principles laws is ensured for the results. In [127], the application of lake temperature prediction is extended to include temporal physical processes. More specifically, a physics-based RNN is developed that involves energy conservation constraints. Standard LSTM models store specific information at each time step, which feeds to the next time step. However, when the models are trained on data from specific seasons or from multiple years, it is difficult to generalize to

data from different time periods since the time profiles vary significantly between each other. By including the energy flux changes, however, which determine the temperature changes, the architecture can successfully predict the lake temperature, even on unseen data. Another example is given in [131], where the data-driven model is penalized with the equation describing the time evolution of waves in order to identify the location of underwater obstacles from acoustic measurements. In this way, the accuracy of the model outside the training dataset is enhanced. Finally, [26] presents a case where multiple physics-based terms are present in a loss function. These might be competing loss terms with multiple local minima and correspond to different physics equations that need to be minimized together. Hence, an approach is presented where the contribution of each term is adaptively tuned during the training phase in order to improve the generalizability of the developed architecture.

2.5.3. Hybrid architecture implementation in computer vision

Integrating useful domain knowledge into DL-based computer vision tasks is essential to build robust, generalizable systems and to compensate for the lack of large-volume training data. An example of such a hybrid architecture is proposed in [252], where the height of a free-falling object is estimated on each frame of a video by training a CNN to detect and track objects obeying to free-falling laws of physics. The training of this CNN is based on a loss function in which first-principles laws are encoded. In [8], physics are blended with DL in the framework of a two-stage encoder with the aim to recover the shape of an object based on polarized photos. In [144] an LSTM architecture is combined with a dynamics model in order to acquire a proposal distribution over an object's state. Finally, in [294], a generative vision system is proposed for estimating physical features of objects by integrating the output of a multi-physics simulation engine in the loop.

Integrating DL techniques with domain knowledge is a recently introduced research topic [133, 216]. As a result, using domain knowledge to derive first-principles models or on a broader perspective, any dynamic mathematical or computer model [173] that describes how the properties of objects and entities change over time (Figure 2.5), and merging them with existing DL architectures constitute an especially promising research task to address the challenges of DL in computer vision.

2.6. Future directions in deep learning for object detection and segmentation in videos

Deep learning has brought a catalytic effect in the field of computer vision for video analysis. Although nobody knows with certainty how DL will evolve over the coming decades, it is expected that much of the future research will revolve around the following critical areas [84, 181, 279]:

- **Out-of-distribution generalization:** Future computer vision systems should be able to make accurate predictions not only in a known context but also for data with different distributions than the ones learned from the training samples. The main reason behind the difficulty of DL systems to accurately generalize

and predict on unseen data is caused by the fundamental assumption that training and test data are independent and identically distributed (IID) [240, 310]. In many real-life cases, however, the IID assumption is hardly satisfied. The ability to generalize under distribution shifts is of critical significance, and hence, the investigation of out-of-distribution generalization is expected to attract enormous research interest in the academic field.

- **Deep learning systems with causal structures:** Causality is expected to be a central strand of DL research in the coming years [206]. Developing DL systems that can represent causal relationships can increase their safety and reliability, and introducing a causal understanding of basic concepts in DL methods could certainly be the key to achieve robustness in complex real-world environments.

- **Effective representation learning with few or no labeled data:** While techniques for representation learning when massive labeled datasets are available have become remarkably powerful, various challenges remain in the case of limited labeled data. Developing approaches for addressing the issue of labeled data scarcity is an emerging popular direction of research.

- **Adaptation in time-varying environments:** Adapting to time-varying environments and other dynamic-behavior-related problems has been under examination for many years and it is expected to gain massive attention by the DL research community over the coming years. Allowing integration of new knowledge online and at the same time being capable of preserving the knowledge learned during previous interactions are only a few of the desirable features of future vision mechanisms.

- **Multi-modal learning:** Ultimately, major emphasis in research is expected to be placed upon developing methods that can process and link information combining modalities from various architectures [149, 180], since unimodal DL methods seem to fail to fulfill all the desirable future DL capabilities. In particular, combined architectures that integrate DL modules with domain knowledge could provide a suitable answer to most research questions arising from the DL directions listed above.

2.7. CONCLUSIONS

In this chapter a study is presented about the detection and segmentation of objects applied to video segments. A review of the currently existing techniques has been presented as well as the major challenges that data-driven techniques face. Then an extension of the data-driven techniques to a hybrid architecture that fuses data-driven techniques with equation-based models describing the dynamic behavior of objects and entities over time has been proposed in order to address issues like data scarcity, generalizability, and interpretability of the purely data-driven architectures. Finally, a survey of the current developments in hybrid architectures has been presented. We hope that this work will assist in better understanding the

current status of DL in computer vision for video analysis as well as in presenting interesting directions as guidelines for future work.

3

A Dataset for Detection and Segmentation of Underwater Marine Debris in Shallow Waters

Efficient marine debris detection is crucial for automating the marine debris collection process. While supervised deep learning achieves state-of-the-art performance in discriminative tasks, replicating this success on underwater data is challenging. The generalization of these methods suffers due to a lack of available annotated data, considering different sources of variation in the unstructured underwater environment and imaging conditions. In this chapter, we present the Seaclear Marine Debris Dataset, the first publicly available shallow-water marine debris dataset annotated for instance segmentation/object detection. The dataset contains 8610 images collected using Remotely Operated Vehicles (ROVs) at multiple locations and with different cameras, annotated for 40 object categories, encompassing not only litter but also observed animals, plants, and robot parts. As part of the technical validation, we provide baseline results for object detection using Faster Region-based Convolutional Neural Network (R-CNN) and You Only Look Once (YOLO)v6 models. Furthermore, we demonstrate the non-triviality of generalizing the trained model performance to unseen sites and cameras due to domain shift. This underscores the value of the presented dataset in further developing robust models for marine debris detection.

This chapter is based on the paper: A. Đuraš, B. J. Wolf, A. Ilioudi, I. Palunko, and B. De Schutter. "A dataset for detection and segmentation of underwater marine debris in shallow waters". In: *Scientific Data* 11.1 (Aug. 2024), p. 921. ISSN: 2052-4463. DOI: 10.1038/s41597-024-03759-2.
Author contributions statement: A. Ilioudi developed the class taxonomy, annotated Marseille data.

3.1. Background & summary

Persistent objects introduced into the marine environment intentionally or unintentionally, as a result of human-induced activities, can be defined as marine debris. Marine debris can injure or even kill marine and coastal wildlife; damage and degrade habitats; interfere with navigational safety; cause economic loss to fishing and maritime industries; degrade the quality of life in coastal communities; and threaten human health and safety [191]. Considering the negative social, economic, and ecological implications associated with pollution, there has been an increase in research dedicated to providing a framework for systematic monitoring and automated collection of marine debris[182]. In this context, two major modalities for detecting marine debris emerged: one focused on surface-floating debris and another dedicated to underwater debris detection. Floating debris detection methods utilize remote sensing technologies such as satellite imagery [137] and aerial photography [122] [291]. To locate underwater marine debris, the development of unmanned vehicles equipped with cameras [235] and acoustic sensors [275] is being pursued to detect debris on the seafloor.

Automated underwater marine debris detection from images shares characteristics and challenges of other automatic recognition vision-based tasks in the underwater domain. Geometric and photometric distortions, introduced to the imaging process by the underwater environment, result in numerous quality-diverse data domains [31]. The visual appearance of aquatic scenes can vary drastically based on the conditions such as depth, turbidity, and type of camera sensors used to obtain the imagery. Thus, obtaining diverse data is crucial for the development and evaluation of robust underwater image processing methods.

Supervised learning methods, which achieve state-of-the-art results on discriminative tasks, e.g. object detection, instance segmentation, depend on the availability of annotated data. In the underwater domain such annotated data is sparse compared to the terrestrial domain, which results in an active research area focused on data augmentation [235] and visual restoration for underwater images [307].

Underwater Marine Debris Detection The only large publicly available repository of underwater marine debris images is the Deep-sea Debris Database (https://www.godac.jamstec.go.jp/dsdebris), curated by the Japan Agency of Marine Earth Science and Technology (JAMSTEC). Available data contains images of marine debris and various types of marine plants and animals captured in underwater surveys by ROVs, mainly in the sea of Japan. The first work on marine debris detection was done by Fulton et al. [65] who annotated a selection of 5720 images from the JAMSTEC database and trained the models for the task of plastic debris detection based on four deep learning architectures for object detection - YOLOv2, Faster R-CNN, Tiny-YOLO, and Single-shot Detector (SSD). The same group of authors extended their previous work by making the *TrashCan* dataset [109] public, increasing the dataset size to 7212 images and providing additional annotations for instance segmentation task as well as more detailed classification of debris by material and instance type. In addition, a more consistent and balanced version of the *TrashCan* dataset named UNO [11] was produced. In this work, the *TrashCan* is further processed to correct wrong label annotations or misplaced bounding boxes,

annotating missing objects, and mitigating category imbalance by fusing all trash categories into one category representing all non-natural objects. The key distinction between our dataset and those derived from the Deep-sea Debris Database lies in the data collection environment. Our data was collected in optically shallow waters (i.e. where light reaches the bottom), consequently making the images susceptible to variations in natural light conditions. Another factor that increases the difference between the appearance of objects in deep water with respect to shallow water and deteriorates the visual conditions is the growth of marine biological fouling on underwater objects. This effect, however, rarely takes place in deep water due to the lack of light. Additionally, our dataset includes data from the same sites but captured with different cameras which can be used to test the generalization of models in the presence of cross-camera domain shift. All mentioned datasets are presented in Table 3.1, along with additional datasets [36] [214] containing land litter and floating marine litter.

The main contributions of the current chapter are the following:

1. We present *Seaclear Marine Debris Dataset* [51] first publicly available underwater marine debris dataset in shallow-water environments, annotated for instance segmentation and object detection tasks. The dataset comprises images gathered from various locations, captured using different cameras, thereby creating a multi-domain dataset.

2. As part of the technical validation, we provide baseline results for marine debris detection with Faster R-CNN and YOLOv6 models.

3. To emphasize the significance of multi-domain data in building robust models, we demonstrate that enhancing objective image quality measures or addressing domain shifts in the input space through image enhancement methods does not improve detection performance or generalization ability in the presence of cross-site or cross-camera domain shifts.

3.2. METHODS

This section provides information about the data acquisition setup, including the robots and camera sensors used to collect raw data for *Seaclear Marine Debris Dataset* [51]. Brief descriptions of each site where data collection was performed are given, outlining the human activity in the area and potential sources of pollution. Additionally, we provide a brief summary of the methods used to analyze and demonstrate the non-trivial nature of performing underwater marine debris detection in multi-domain settings.

Degradations introduced by turbidity, floating particles, and the properties of light propagation in water, typically result in images of low visual quality i.e., with color distortion, contrast decrease, and haziness. In addition to camera and medium dependencies, shallow-water images have a strong dependency on natural lighting, which can result in drastically different images for the same site depending on capture time and weather conditions. Low visual quality and domain shift can impair

Table 3.1: Overview of annotated image datasets for underwater marine debris detection

Dataset	Environment	No. images	Annotation type	No. categories	Year
Trash-ICRA19 [66]	Underwater (Plastic, ROV, bio)	5720	Bounding Box	3	2019
TrashCan-Material 1.0 [110]	Underwater (Plastic, metal, paper, rubber, wood, etc.)	7212	Mask/Bounding Box	16	2020
TrashCan-Instance 1.0 [110]	Underwater (Bag, clothing, rope, wreckage, etc.)	7212	Mask/Bounding Box	22	2020
UNO [11]	Underwater (Bag, clothing, rope, wreckage, etc.)	5902	Bounding Box	4	2022
PlastOPol [37]	Land/floating (Litter)	2418	Bounding Box	1	2022
DeepPlastic [262]	Underwater (Plastic litter)	3200	Bounding Box	1	2021
TACO [214]	Land/floating (Cigarette, plastic film, broken glass, styrofoam piece, etc.)	1500	Mask/Bounding Box	60	2020
CleanSea [234]	Underwater (Litter)	1223	Mask/Bounding Box	19	2022

the generalization ability for object detection using deep learning architectures [31]. These issues are typically addressed by image processing techniques categorized as image restoration methods, assuming a known image formation and degradation model, or image enhancement methods that use subjective quality criteria to produce visually pleasing images. Model-based image restoration methods used in underwater environments typically depend on parameters that vary depending on water type, depth, lighting, and camera parameters. The data available within the *Seaclear Marine Debris Dataset* is collected from multiple trials at different sites and captured with cameras of different characteristics. Thus estimating these parameters would be difficult if not impossible without performing a calibration procedure in situ for each conducted survey. Since image restoration is not applicable, in this chapter we resort to image enhancement techniques that only require a single image as an input.

First, we describe Underwater Image Quality Measure (UIQM) and Underwater Index, quality assessment metrics used to gain insight into the visual characteristics of each domain. Finally, this is followed by a description of the fusion-based image enhancement method [4] which we use as a preprocessing step to improve the visual characteristics of the data and reduce the domain shift.

3.2.1. DATA ACQUISITION

Data collection was performed by deploying camera-equipped ROVs at different sites. A *BlueROV2* was equipped with two cameras, a *Bluerobotics Low-Light HD Camera* and a *Paralenz Vaquita*, while the *SST Mini-Tortuga ROV* was equipped with a *Smart Security SIP-E323CV* camera. Through multiple trials, different lighting and turbidity conditions were encompassed, with each site having a characteristic source of pollution.

- *Portoč (Island of Lokrum), Croatia* - situated 600 meters away from the city of Dubrovnik, is used as a small port for docking of tourist ferries. Data includes footage of debris accumulated mostly as a result of tourist activity together with the vicinity of cruise ship and yacht anchorage. Moreover, some of the images collected from the Lokrum site contain debris that was intentionally placed on the seabed and in the water column by divers for Seaclear project demonstrations (an example can be seen in the LO-II labeled image in Figure 3.7). The Lokrum site features clear transparent water and *Posidonia Oceanica* seabeds at $5-10$ m depth.

- *Bistrina, Croatia* - situated in Mali Ston Bay, the largest production area of the European Flat Oyster (*Ostrea edulis*). Oyster farming activities commonly result in the marine environment being polluted with debris such as plastic shellfish trays, buoys, and nets, which is reflected in the imagery for this site, as seen in Figure 3.7. The water is turbid, exhibiting lower visibility due to increased nutrient and sediment concentrations.

- *Slano, Croatia* - a small village with a harbor located 27 km northwest of Dubrovnik. Pollution is mostly the result of harbor and tourist activities along

with improper disposal of construction waste. The debris is diverse and clustered, partially covered in construction rubble, which causes it to blend in with the background.

- *L'Estaque (Marseille), France* - suburb area located north of Marseille, in the vicinity of the old port. Most of the debris is the result of the industrial activity and waste dumping.
- *Jakljan, Croatia* - islet belonging to the Elaphites archipelago. The data features a smaller number of individual pieces of debris, mostly bottles and cans as a result of nautical tourism.

3.2.2. UNDERWATER IMAGE QUALITY ASSESSMENT

The available quality metrics for terrestrial color images are limited in their applicability to underwater images since they fail to consider the extent of degradation and optical properties involved in underwater image formation. Since no reference image is available and subjective measures require time-consuming manual labeling, objective underwater quality measures that aim to capture the objectivity and perception of the Human Visual System (HVS) are utilized to provide a quality estimation on the *SeaClear Marine Debris Dataset*.

In literature [32] it was observed on large amounts of underwater image data that the Lab color space has a strong capability of indicating the color distribution and that its (a, b) components can be used to differentiate between underwater and terrestrial images. Underwater images typically gather further away from the origin, while terrestrial images are usually distributed sparsely around it, which allows formulating the score called Underwater Index [32] U representing the possibility of the image being taken underwater:

$$U = \frac{\sqrt{d_o}}{10\bar{L}d_a d_b} \tag{3.1}$$

where $\bar{L}$ is an average value of the L channel, while d_o, d_a, and d_b are distances from the origin, along the a axis, and along the b axis, respectively.

UIQM [200] is composed of Underwater Image Colorfulness Measure (UICM), Underwater Image Sharpness Measure (UISM), and Underwater Image contrast Measure (UIConM), representing a comprehensive quality of an underwater image, where its sub-indexes evaluate colorfulness, sharpness, and contrast characteristics, respectively.

UICM is calculated in terms of the variance σ^2 and the mean μ of the opponent color components:

$$\mathrm{RG} = R - G \qquad \mathrm{YB} = \frac{R+G}{2} - B \tag{3.2}$$

where asymmetric alpha-trimmed statistics $\hat{\sigma}^2$ and $\hat{\mu}$ are used to avoid the effect of outlier intensities on the measure:

$$\mathrm{UICM} = -0.0268\sqrt{\hat{\sigma}^2_{RG} + \hat{\sigma}^2_{YB}} + 0.1586\sqrt{\hat{\mu}_{RG} + \hat{\mu}_{YB}} \tag{3.3}$$

To account for the sharpness loss, first an edge image is obtained by applying the Sobel-Feldman operator $S(\cdot)$ [246] on each color channel of the input image $\mathbf{I}$. UISM is then defined as a linear combination based on the enhancement measure estimation function [201] $\mathscr{E}(\cdot)$ measuring the contrast ratio of blocks in each edge image, as follows:

$$\text{UISM} = 0.299 \cdot \mathscr{E}(S(\mathbf{I}_R)) + 0.584 \cdot \mathscr{E}(S(\mathbf{I}_G)) + 0.114 \cdot \mathscr{E}(S(\mathbf{I}_B))$$

where the weights for each channel are chosen to reflect the response of the human visual system.

The final component accounts for the contrast degradation typically caused by backward scattering using the logAMEE measure on the intensity image. logAMEE combines the logarithmic entropy of the Michelson Contrast and PLIP operators ($\otimes$, $\oplus$, $\ominus$), which provide nonlinear representation consistent with HVS:

$$\text{UIConM} = \frac{1}{L \cdot M} \otimes \sum_{l=1}^{L} \sum_{m=1}^{M} \mathscr{M}(W_{l,m}) \ln(\mathscr{M}(W_{l,m})) \tag{3.4}$$

where $L \cdot M$ is the number of blocks and $\mathscr{M}(\cdot)$ is the PLIP Michelson Contrast [153].

The final value of the UIQM [200] is given by:

$$\text{UIQM} = c_1 \cdot \text{UICM} + c_2 \cdot \text{UISM} + c_3 \cdot \text{UIConM} \tag{3.5}$$

where the choice of the coefficients c_1, c_2, and c_3 depends on the application, but generally a higher value of UIQM value corresponds to an image with a better quality.

3.2.3. FUSION BASED IMAGE ENHANCEMENT

The Fusion-based image enhancement method [4] is a framework based on blending filters frequently used for image enhancement. This method can be summarized with 3 steps:

1. Obtaining classical enhanced versions of a degraded image
2. Deriving weight maps for each enhanced version
3. Multi-scale fusion using the weight maps

The two derived enhanced versions represent the color-corrected version of the image and the contrast-enhanced version of the underwater image after noise reduction, respectively. These two enhanced versions are referred to as inputs for the Fusion algorithm. The first input is obtained by applying a white balancing technique, which improves the Gray-World illumination estimate by adding a weighting based on the size of the detected set of colors. To remove degradation caused by the scattering in the medium, the second input is derived by applying the local adaptive histogram equalization to a noise-free and color-corrected version of the original image.

Fusion enhancement utilizes 4 weight maps per input to enhance the contrast, saturation, and exposedness of the image. The Laplacian contrast weight (W_L) tackles the issue of global contrast by applying a Laplacian filter to each luminance channel of the input and calculating the absolute value of the filter's output. The local contrast weight (W_{LC}) considers the relationship between each pixel and the average of its neighboring pixels. This measure enhances the visual effect of local contrast by emphasizing transitions, particularly in the highlighted and shadowed areas of the second input. The saliency weight (W_S) enhances the visibility of objects that lose their distinctiveness by applying a saliency algorithm based on the biological concept of center-surround contrast. Additionally, to avoid prioritizing highlighted areas in W_S and to protect the mid tones that might be altered in some specific cases, the exposedness weight (W_E) is defined. Exploiting the fact that pixels tend to have a higher exposed appearance when their average normalized values are closer to 0.5, W_E is defined as the Gaussian distance to the 0.5 value. This has the effect of tempering the saliency weight and preserving image appearance for non-highlighted areas.

The final enhanced image version is obtained by fusing the defined inputs and weights at multiple scales. Inputs I^k are decomposed into a pyramid by applying the Laplacian operator $L(\cdot)$ to different scales. The weight maps for the k-th input are normalized to satisfy the constraint $\sum \bar{W}^k = 1$. For each normalized weight map $\bar{W}$ a Gaussian pyramid $G(\cdot)$ is computed. Since both the Laplacian and Gaussian pyramids have the same number of levels l, mixing is performed independently for each level. Finally, to compute the enhanced image, we use:

$$E(x,y) = \sum_{k=1}^{K} G^l\{\bar{W}^k(x,y)\} L^l\{I^k(x,y)\}$$

Several examples of Fusion-enhanced images are shown in Figure 3.4.

3.3. DATA RECORDS

This section describes the image data included in the *Seaclear Marine Debris Dataset*. Taxonomy of debris categories and debris instance distribution are visualized and followed by a brief overview of the annotation format and directory structure of the dataset. The dataset is made publicly available at the 4TU.ResearchData repository [51] under the CC BY 4.0 license.

DATASET STRUCTURE

The *Seaclear Marine Debris Dataset* is comprised of 8610 underwater marine debris images, captured utilizing BlueROV and Mini-Tortuga ROVs and annotated for instance segmentation/object detection tasks. Object instances annotated in this dataset can be semantically grouped into 3 super-categories:

- *debris* - objects found in the marine environment as a result of human-induced activities,

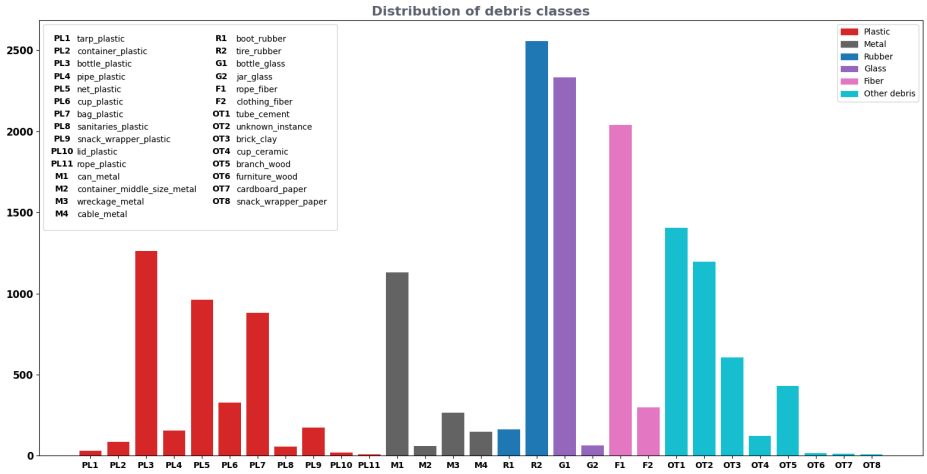


Figure 3.1: Distribution of debris instances in the dataset images by object categories and materials.

- *bio* - marine vegetation and animals,
- *robot* - ROVs used for data collection and their parts.

Debris annotations are categorized by instance type and material, both encoded in the category name as *{instance}_{material}*. The taxonomy of the class categories

Table 3.2: No. images per dataset domains, grouped by data collection sites and cameras.

Site	Camera	Domain Label	No. images
Bistrina, Croatia	Bluerobotics Low-Light	BIS-I	1390
	Paralenz Vaquita Gen 2	BIS-II	2069
	SIP-E323CV	BIS-III	193
Jakljan, Croatia	Bluerobotics Low-Light	JA-I	241
	Paralenz Vaquita	JA-II	65
Lokrum, Croatia	Bluerobotics Low-Light	LO-I	556
	Paralenz Vaquita Gen 2	LO-II	77
	SIP-E323CV	LO-III	339
Marseille, France	SIP-E323CV	MS-I	3441
Slano, Croatia	Bluerobotics Low-Light	SL-I	168
	Paralenz Vaquita	SL-II	71
		Total	**8610**

is depicted in Figure 3.2, while the distribution of the debris categories included in the dataset is shown in Figure 3.1. For the *bio* category, the animal species was indicated in the annotations, while for the *robot* category the model of the ROV and specific robot parts such as cable or vehicle leg which frequently appeared in the images, were indicated.

Images were labeled using *labelme* [230], a tool that allows creating polygon annotations for instance segmentation and that provides easy conversion from the *labelme* JSON format to the frequently used Visual Object Classes (VOC) and Microsoft Common Objects in Context (COCO) formats. The dataset directory structure reflects the site at which data was captured and the camera used to obtain the images as follows:


```
Seaclear Marine Debris Dataset
├── Bistrina
│   ├── Bluerobotics HD
│   ├── Paralenz Vaquita Gen 2
│   └── SIP-E323CV
├── Jakljan
│   ├── Bluerobotics HD
│   └── Paralenz Vaquita
├── Lokrum
│   ├── Bluerobotics HD
│   ├── Paralenz Vaquita Gen 2
│   └── SIP-E323CV
└── Marseille
```

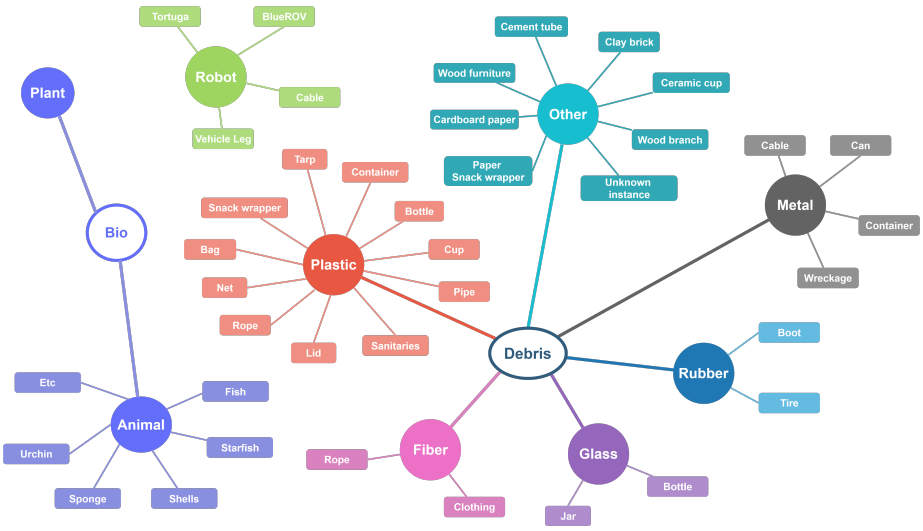


Figure 3.2: Visualization of the relations between the debris categories and super-categories.

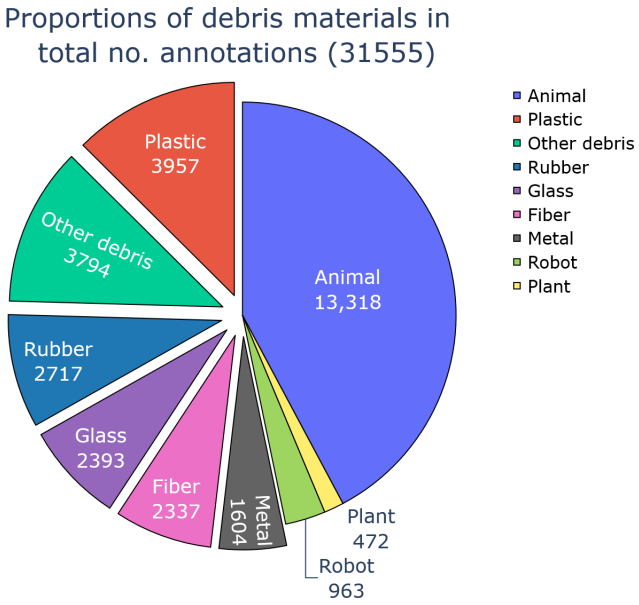


Figure 3.3: A pie chart showing the proportions of annotated objects belonging to specific debris material, robot, animal and bio categories.

```
    └─ SIP-E323CV
└─ Slano
    ├─ Paralenz Vaquita
    └─ Bluerobotics HD
```

Information on the number of images and annotations for each sub-folder are provided in Table 3.2. Visualized samples of annotated images from the dataset can be seen in Figure 3.7.

3.4. TECHNICAL VALIDATION

To enable new research and to provide better insight into the characteristics of the provided data and open challenges, the following experiments were conducted as part of this work:

1. Baseline results on combined data from all sites in *Seaclear Marine Debris Dataset* [51] for the debris detection task are given with both a two-stage Faster R-CNN and a one-stage YOLOv6 detector to provide a reference for future work.

2. Analysis of domain effect on object detection performance in shallow-water imagery is conducted based on cross-camera and cross-site data splits. By comparing performance on source and fusion-enhanced data we investigate

the effects of image enhancement on improving the generalization ability of trained models.

The results are summarized in Table 3.4 and Table 3.5. The experiments and the results are preceded by a brief overview of the architectures and design choices for Faster R-CNN and YOLO V6 models used to conduct the experiments as part of the technical validation.

3.4.1. BASELINE MODELS

For obtaining the baseline results on the *SeaClear Marine Debris Dataset* for marine debris detection task we used the Faster R-CNN and YOLOv6 [155] models, which we briefly review in this section. Faster R-CNN [223] is a two-stage model and more computationally expensive; on the other hand, YOLOv6 is a lightweight model suited for real-time inference.

Table 3.3: Per domain UIQM and Underwater Index values, for source and fusion enhanced data.

	UIQM		Underwater Index	
	Source	**Fusion Enhanced**	**Source**	**Fusion Enhanced**
BIS-I	1.544	3.358 ↑	13.932	2.246 ↓
BIS-II	1.978	3.507 ↑	22.672	2.122 ↓
BIS-III	1.898	3.212 ↑	6.106	2.085 ↓
JA-I	2.692	3.426 ↑	5.024	2.941 ↓
JA-II	2.252	3.490 ↑	2.376	1.949 ↓
LO-I	2.279	3.274 ↑	2.768	3.104 ↑
LO-II	1.991	3.443 ↑	4.777	1.496 ↓
LO-III	1.598	3.136 ↑	3.496	3.558 ↑
MS-I	1.606	3.291 ↑	7.345	1.878 ↓
SL-I	2.928	3.416 ↑	8.209	4.762 ↓
SL-II	2.084	3.447 ↑	3.985	2.192 ↓

Table 3.4: Stratified split baseline results.

Model	**Fusion Enhancement**	**mAP (%)**	**mAR (%)**	**mAP (%) per site**				
				Bistrina	**Jakljan**	**Lokrum**	**Marseille**	**Slano**
Faster R-CNN R50 + FPN		61.7	68.1	60.7	58.2	55.9	80.4	32.6
Faster R-CNN R50 + FPN	✓	59.0	65.6	57.2	51.9	52.4	79.2	28.7
YOLOv6 S		68.3	75.0	69.5	66.7	59.3	79.0	46.9
YOLOv6 S	✓	**68.9**	**75.4**	69.6	70.2	60.3	80.6	46.2

Table 3.5: Evaluation of generalization performance in cross-camera and cross-site setting

Model	Fusion Enhancement	Split		mAP (%)	mAR (%)
		Train	Test		
Cross-camera Domain Generalization					
Faster R-CNN R50 + FPN		BIS-II	BIS-I	14.5	22.0
Faster R-CNN R50 + FPN	✓	BIS-II	BIS-I	14.21	23.45
YOLOv6 S		BIS-II	BIS-I	**26.2**	**46.6**
YOLOv6 S	✓	BIS-II	BIS-I	24.4	44.5
Faster R-CNN R50 + FPN		BIS-I	BIS-II	12.0	18.0
Faster R-CNN R50 + FPN	✓	BIS-I	BIS-II	14.16	21.41
YOLOv6 S		BIS-I	BIS-II	24.7	38.8
YOLOv6 S	✓	BIS-I	BIS-II	**26.0**	**40.4**
Cross-site Domain Generalization					
Faster R-CNN R50 + FPN		{BIS-*}	{LO-*}	5.0	9.7
Faster R-CNN R50 + FPN	✓	{BIS-*}	{LO-*}	3.7	8.3
YOLOv6 S		{BIS-*}	{LO-*}	**20.4**	**51.1**
YOLOv6 S	✓	{BIS-*}	{LO-*}	7.2	42.7

FASTER R-CNN

The Faster R-CNN architecture can be divided into the following sub-modules:

- **Backbone Network** is a Convolutional Neural Network (CNN) used to extract features from the input image (e.g. ResNet, VGG, AlexNet), typically pre-trained on a large dataset such as ImageNet. The choice of the backbone model determines the number of model parameters and the representation ability of the model.

- **Region Proposal Network (RPN)**: This is the first stage of the model that generates a set of region proposals, or regions of interest (RoIs), which are image areas likely to contain objects. The RPN is a lightweight CNN that is trained to predict the objectness score and to regress the offsets for a fixed number of anchor boxes at each spatial location.

- **RoI Pooling** is applied to the feature maps generated by the backbone network, it converts the generated RoIs into fixed-size feature maps, which can then be fed into the second stage of the model.

- **Fast R-CNN Head**: The second stage of the model consists of fully connected layers that produce the final output by predicting class probabilities and bounding box coordinates.

YOLOv6

As in the Faster R-CNN architecture, YOLOv6 uses a CNN backbone network to extract features from the input image; however, since YOLOv6 is a single-stage detector, it predicts the class probabilities and the bounding box without explicitly generating regions of interest. Single-stage detectors generally consist of two additional submodules: a neck and a head. The neck of the network is used for aggregation of low-level spatial features and high-level semantic features, which are used by the head to produce the final detection results. Design choices for the YOLOv6 submodules are made aiming to improve the computational efficiency and accuracy trade-off when compared to its predecessors, specifically YOLOv5 and YOLOX models.

The EfficientRep backbone used in YOLOv6 utilizes the re-parametrization strategy inspired by RepVGG [48] to decouple the multi-branch topology at learning time and to provide a simpler single path topology at inference time. The Rep-PAN neck is a modified version of the PAN topology from YOLOv4 and YOLOv5 models that replaces the CSP-Block with a RepBlock for small models and CSPStackRep for large models. Unlike YOLOv5 the design of the YOLOv6 detection head decouples the classification and regression layers like YOLOX, but reduces the number of convolution layers to increase efficiency.

3.4.2. Experimental setup

Baseline results were obtained using a conventional approach to object detection. A two-stage Faster R-CNN and a one-stage YOLOv6 model initialized with COCO dataset weights were finetuned to the *Seaclear Marine Debris Dataset*. We used a larger (41.6 M parameters) FasterR-CNN model, utilizing a Resnet-50 backbone with FPN for feature extraction and a smaller YOLOV6 S model (17.2 M parameters).

Faster R-CNN training was performed for 150 epochs using the cosine annealing (with warm restarts) learning policy with a base learning rate of $\lambda_0 = 0.001$ and $T_0 = 10$ steps, with a multiplication factor of 2. Reported results were obtained using 640×360 input images and mini-batches of size 2. For YOLOv6 S finetuning was done based on the implementation available in the official repository. The model was fine-tuned for 400 epochs with a batch size of 32 and an input image size of 640 × 360.

To mitigate the impact of categories with a small number of annotations we consolidated categories in the dataset that had fewer than 50 annotations by grouping them with appropriate, related categories. For example, *snack_wrapper_plastic* (8 annotations) and *snack_wrapper_paper* (172 annotations) were combined into *snack_wrapper* (180 annotations) category. This process resulted in the final dataset of 34 categories used to conduct the experiments presented in further text.

3.4.3. Baseline results

To obtain the training and test set for the baseline evaluation we performed an 80 %-20 % random split on the whole dataset. Both models exhibited high performance in this setting with YOLOv6 S outperforming the Faster R-CNN model by ≈ 7 %.



Figure 3.4: The top row showcases the original images from different data collection sites of our dataset, while the bottom row displays the corresponding Fusion-enhanced versions. A more uniform color distribution can be observed in the Fusion-enhanced images.

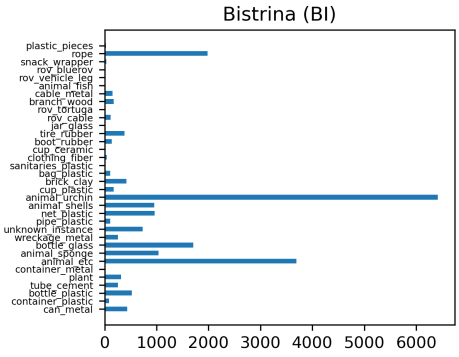
This is most likely due to better regularization of the smaller model on a dataset of this size and YOLOv6 using multiple data augmentation strategies, which were not utilized on Faster R-CNN. Performance was consistent for both models across sites, as seen in Table 3.4, with both models showing the highest performance on the Marseille data, most likely due to the lower inter-class variance and the static nature in which the data was collected. Also, both models exhibit the lowest performance on the Slano data, which is the most challenging site in our dataset due to the small sample size and the large amounts of various types of debris.

Additionally, we evaluated both models on baseline data split with fusion-enhanced images. For Faster R-CNN, we report a 2.7% performance drop in terms of mean Average Precision (mAP), compared to evaluation on unprocessed data; however, on YOLOV6, a slight improvement of 0.5 % was achieved.

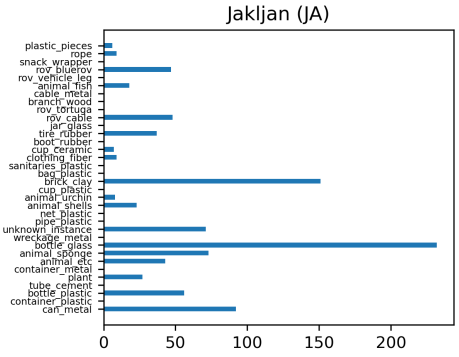
EFFECTS OF IMAGE ENHANCEMENT AND EVALUATING GENERALIZATION

As seen in Table 3.3 processing the images with Fusion-based image enhancement seems to provide a two-fold benefit by significantly improving the image quality as measured by UIQM and by lowering the Underwater Index. Visualizing the enhanced images versus the source images in terms of (a, b) components of Lab color space (see Figure 3.8), shows that this sort of processing eliminates the color distribution gap between different domains in the image input space. However, these improvements in terms of objective quality metrics do not correspond to consistent improvement of object detection performance nor a better generalization ability of the model. In a cross-site setting, both models perform poorly and no performance improvements are observed from applying image enhancement.

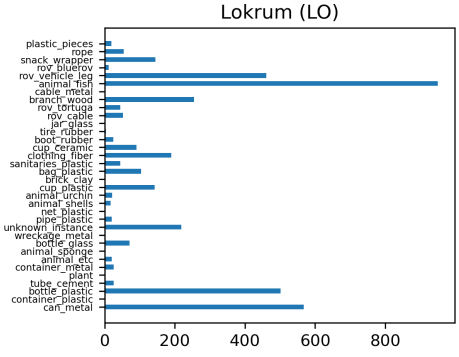
Evaluating in a cross-camera setting, where the model was trained on BIS-I data and evaluated on BIS-II data we observed an improvement of 2.1% in terms of mAP. However, this improvement does not hold for the inverse case where the model is trained on the BIS-II data and evaluated on BIS-I data. It is important to note that for these two domains, the shift was caused only by camera pose offset and camera type since both were mounted on the same ROV during the data collection survey. This results in a minimal difference in terms of category distribution between



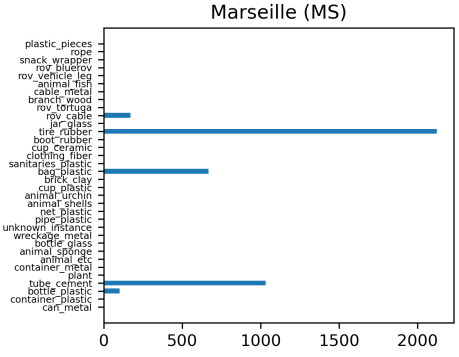
(a) Bistrina (BI)



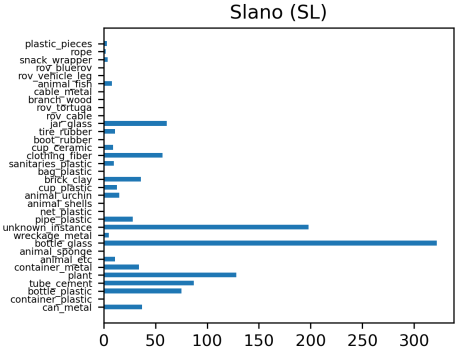
(b) Jakljan (JA)



(c) Lokrum (LO)

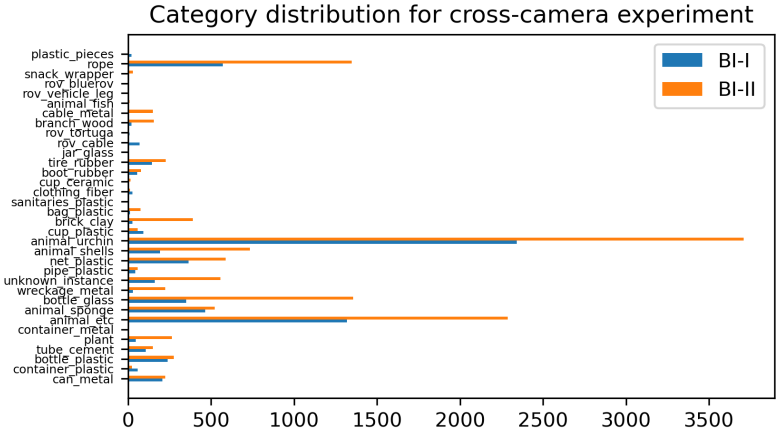


(d) Marseille (MS)

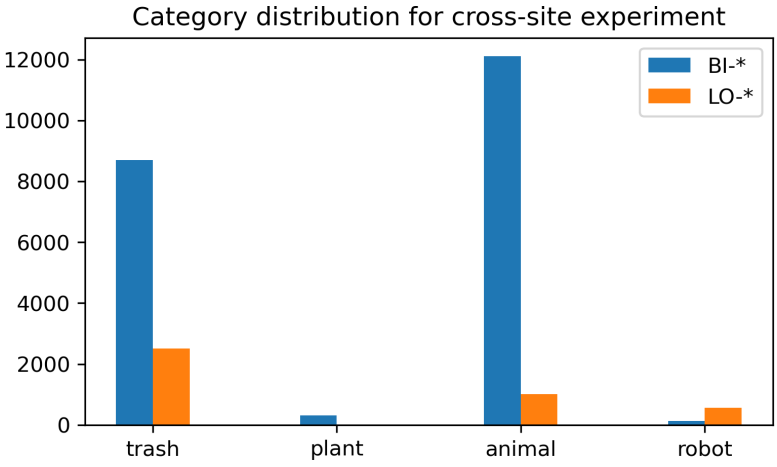


(e) Slano (SL)

Figure 3.5: Distribution of object categories for each site used in the evaluation of baseline results.



(a)



(b)

Figure 3.6: Category distributions for domains used in cross-camera and cross-site experiments.

training and validation data as seen in Figure 3.6. However, there is still a significant drop in performance compared to the baseline results where data from all domains is readily available. This suggests that a significant part of the domain shift can be attributed to using different cameras and the changes in point of view from how each camera was mounted.

In a cross-site setting, imbalanced category distributions between training and validation data are unavoidable because the marine life and type/quantity of debris depend on specific locations and human activity. This issue is exaggerated by the large number of categories as objects are less likely to belong to the same category. Using the full 34 categories of the dataset was not viable in the cross-site setting as there was little overlap between the category distributions. In an attempt to produce a more balanced distribution, two strategies were explored: aggregating debris categories based on materials (plastic, metal, rubber, fiber, glass, and other debris) as previously done in literature [65] and consolidating all debris categories into a single category. Non-debris categories were grouped into one of the appropriate categories: *robot*, *animal* or *plant*. Data from the Bistrina site is used as training data because it provides the most diverse domain. Lokrum site was chosen for validation data because it has data available from the same set of cameras as the training domain, allowing us to isolate the site effect on the domain shift and evaluate its impact on performance. As demonstrated in Table 3.4, the performance in terms of mAP degrades further in cross-site scenarios. This outcome is expected, as using different sites for training and validation significantly affects the variance in category distribution and increases intra-category variance. An outlier among the cross-site experiments is the top-performing one, reporting a mAP of 20.4%. Upon inspecting the AP per category, it was observed that this result was driven by the model's high average precision (AP) for the robot category and not by improved debris detection performance.

In addition, we observe in several cases, especially in the cross-site and cross-camera scenarios reported in Table 3.5, that detection performance is worse with enhanced data. This paradox arises because enhancement is optimized for perceptual quality rather than task-specific relevance. While it improves sharpness, contrast, and color balance (Figure 3.4), these changes might suppress or distort visual cues that are important for object detection, such as subtle boundaries or material textures. In cross-site settings, where debris types and background conditions differ significantly, enhancement may smooth away these distinguishing characteristics, reducing the model's ability to adapt. Similarly, in cross-camera settings, enhancement may introduce inconsistent visual artifacts across domains due to differences in camera noise or lens characteristics, which further reduces generalization performance. These effects are especially complex in underwater scenarios where object appearance is already highly variable and context-dependent.

We report the mAP in Table 3.5 using the full 34 categories of the dataset for cross-camera experiments and using the 4 aggregated categories for cross-site experiments.

USAGE NOTES

Annotations and additional information about the dataset are stored in COCO (Common Objects In Context) style format (single JSON file). Using this standard annotations format allows utilizing open-source tools, like COCO API (https://github.com/cocodataset/cocoapi) for easy access through multiple programming languages (Matlab, Lua, Python). More complex utility functions for visualizing and modifying annotations are available in open-source Python packages, such as KWCOCO (https://github.com/Kitware/kwcoco).

CODE AVAILABILITY

The custom code used for generating figures, conducting image quality analysis, and image enhancement can be found in the repository associated with this publication (https://github.com/adjuras/seaclear-dataset). Our MATLAB code used for Fusion-based image enhancement is a slightly modified version of an open-source implementation available online (https://github.com/fergaletto/Color-Balance-and-fusion-for-underwater-image-enhancement.-.).

Official UIQM implementation used to obtain values presented in Table 3.3 is available for download per request on the author's website (https://karen-panetta.squarespace.com/download). YOLOv6 S implementation used for the technical validation is available in the official repository (https://github.com/meituan/YOLOv6).

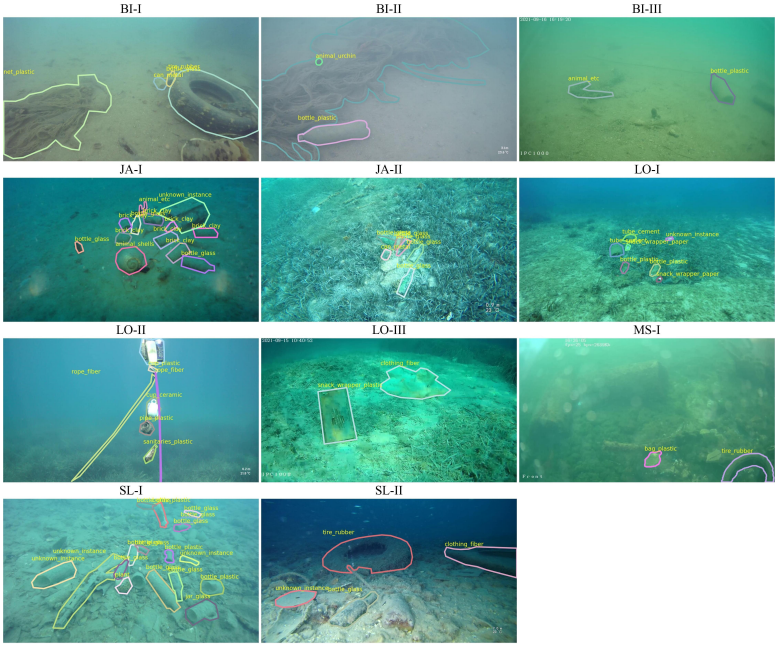


Figure 3.7: Annotated sample visualization from each domain of the dataset, showing both polygon masks and category labels

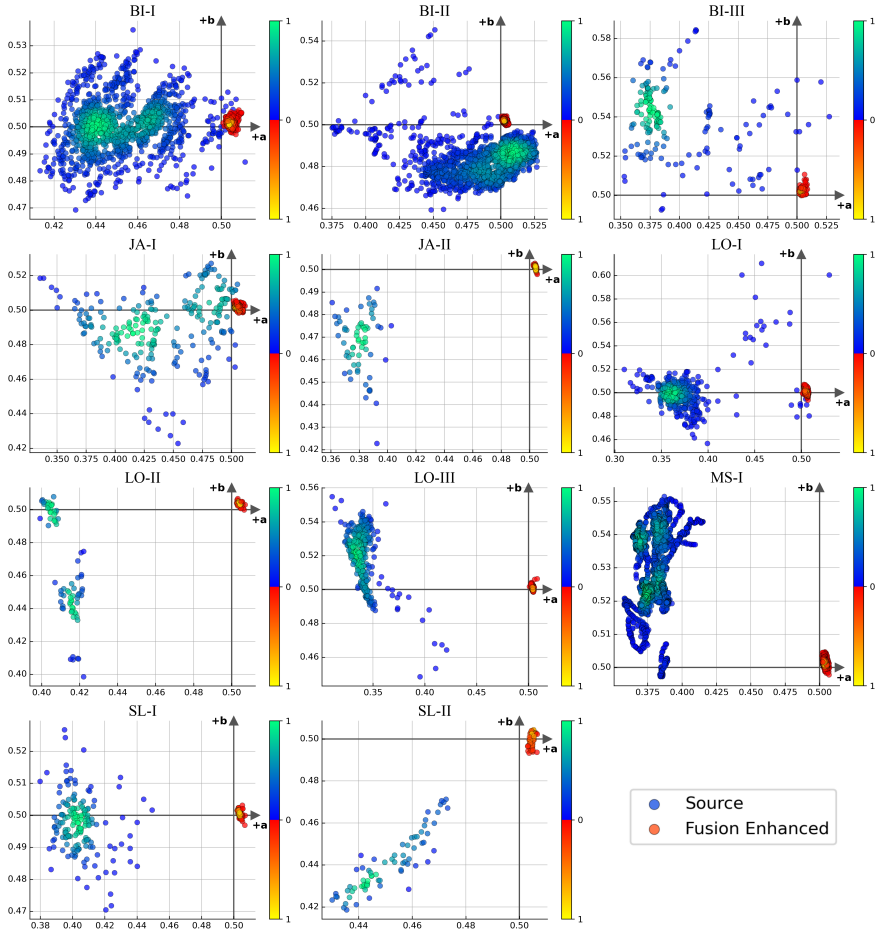


Figure 3.8: Scatter plots of source and enhanced data per domain in terms of (a, b) components in lab color space, showing the distinct grouping of the Fusion-enhanced data around the origin.

4

The SeaClear System: An Intelligent Multi-Robot Solution for Autonomous Cleanup of Marine Debris on the Seabed

Marine debris poses an alarming threat to ocean environments. Conventional methods of sea and ocean cleaning rely heavily on manual collection, a process that has repeatedly demonstrated its inefficiency and extensive demand for resources. This chapter presents the SeaClear system, a novel multi-robot platform designed to autonomously detect and collect marine debris, thereby offering a more efficient solution to this environmental challenge. An overview of the system is presented, followed by a detailed description of each robot's capabilities. Leveraging artificial intelligence, the system employs the deep-learning-based computer vision algorithm You Only Look Once (YOLO) for the detection of underwater litter, addressing the challenges of poor visibility and hydrodynamic disturbances of underwater environments. Additionally, the chapter explores the implemented navigation and control methodologies, which are an essential part of the workflow of the system. The performance of the designed system is validated via field tests conducted in a real-world underwater environment. Finally, directions for future work are proposed.

This chapter is based on the paper: A. Ilioudi, S. Sosnowski, E. Banken, P. Bevanda, J. Brüdigam, L. Buşoniu, Y. Chardard, C. Delea, B. De Schutter, A. Đuraš, C. Hertel-ten Eikelder, S. Heshmati-Alamdari, V.-M. Maer, I. Palunko, I. Pozniak, V. Prkačin, and D. Tolić. "The SeaClear system: an intelligent multi-robot solution for autonomous cleanup of marine debris on the seabed". In: *Engineering Applications of Artificial Intelligence* (May 2024). Submitted, under review.

Author contributions statement: A. Ilioudi developed the deep-learning-based underwater debris detection module, conducted and analyzed the validation experiments for the litter detection component, structured, and co-wrote the manuscript.

4.1. Introduction

Marine environments are increasingly threatened by the accumulation of waste resulting from human activities. Plastic is the largest and most harmful fraction of marine waste, accounting for around 85% of total marine waste [186] [119]. It is estimated that 11 million tons of plastic waste enter the ocean each year [186]. Interestingly, the majority of plastic entering the oceans (94%) ends up on the seafloor instead of the sea surface. The severity of the situation is underscored by multiple studies warning that the volume of marine debris may continue to increase, potentially surpassing the total weight of fish in the sea by 2050 if no drastic action is taken [292].

Parallel to this environmental threat, the rapid advancements in artificial intelligence and robotics have led to their extensive application outside the lab for complex, real-world problems. Despite their potential though, the implementation of these technologies for addressing the urgent issue of marine pollution, particularly litter on the seabed, remains a challenging and largely unexplored area. The primary reasons hindering the employment of subsea robots for debris collection are related to the nature of underwater environments. Disturbances caused by hydrodynamic forces and external factors can heavily affect the performance of robotic vehicles, especially in cases that require high operating precision. Another major challenge is related to the poor visibility conditions underwater due to light absorption, scattering, and reflection. Finally, underwater communication and sensing are imposing limitations on localization and control in the field of underwater robotics.

Artificial Intelligence (AI) adoption in underwater robotics applications could play a pivotal role in addressing the issue of marine litter. This chapter introduces the SeaClear system, an autonomous multi-robot platform engineered for the collection of marine debris from the seafloor. This system stands as a significant application of AI in the field of marine robotics, offering a novel solution to an increasingly concerning environmental problem. Our objective is to explain the key features of the SeaClear system and workflow, as well as to validate its feasibility and practical performance through results from field experiments. Specifically, the contributions of the current chapter are:

- The infrastructure of the SeaClear system is described, consisting of a multi-robot solution that integrates aerial, surface, and underwater vehicles. This system is designed for the efficient and autonomous detection, mapping, and collection of marine debris from the seafloor.

- The application of the state-of-the-art computer vision algorithm YOLO for the detection and classification of underwater litter is detailed. The adaptability and performance of this algorithm under the challenging conditions of underwater environments are assessed, providing valuable insights from its deployment in real-world scenarios.

- A comprehensive sensing and control scheme is developed, that facilitates the seamless integration and operation of the multi-robotic platform.



Figure 4.1: Overview of the SeaClear robotic platform

- The SeaClear system is validated through various field experiments, confirming its practical performance and revealing key insights.

This research provides a foundational blueprint for the development and extension of underwater multi-robot autonomous systems, incorporating sophisticated workflows and AI applications designed for marine ecosystem protection.

The chapter is organized as follows: Section 4.2 gives an overview of the existing related work in this field. Section 4.3 presents the overall architecture of the platform and an in-depth description of each robot constituting the system. Section 4.4 describes the software architecture of the SeaClear system. In Section 4.5 the proposed workflow of the SeaClear system is presented and analyzed. Section 4.6 presents the results of the simulation validation and the field experiments. Finally, Section 4.7 gives concluding remarks and directions for future work.

4.2. RELATED WORK

Despite the growing threat posed by marine litter, there are only a few integrated solutions to address it. The majority of the current strategies primarily focus on preventive measures and pollution monitoring [9, 286], with the cleanup approaches typically targeting the sea's surface and the water column. In [39] the first global map of plastic waste in the ocean is presented. An approach to detect and identify submerged debris in marine environments using forward-looking sonar imagery and convolutional neural networks is presented in [275]. In [65] various deep-learning algorithms performing the task of visually detecting trash in underwater environments are evaluated. However, all these solutions only focus on

the detection and mapping of the litter, without proposing an approach for litter collection. The employment of robotics to facilitate comprehensive solutions to the identification, gathering, and elimination of marine waste is largely concentrated on the collection of surface marine litter. These efforts include, e.g., the Ocean Cleanup [264] initiative, which has developed an approach based on natural, oceanic currents to passively collect plastic debris in floating garbage patches. More specifically, their system comprises a floating barrier for capturing waste and a central platform that functions as a processing plant to extract, sort, and recycle the collected plastic. In addition, in [143] a robot system is developed to collect floating garbage from the water surface. A deep-learning-based detection framework is adopted for the floating litter detection, while a sliding-mode control approach is implemented for the steering of the robots. In [308] an unmanned surface vehicle is proposed for autonomous water quality monitoring as well as for water surface litter detection and collection. A color-based object detection algorithm is employed for the detection of floating litter in combination with sensor fusion to accomplish the various functions of water quality monitoring and litter cleaning. Finally, a net is mounted to the surface vehicle to collect the detected floating litter.

Furthermore, the organization SeaCleaners [266] has developed a specifically designed vessel to collect floating plastic debris. This vessel is also capable of removing larger pieces of floating debris from the water, such as nets or containers lost by other vessels, using two cranes. In addition, Ranmarine [219] has introduced an autonomous lightweight vessel called Watershark, inspired by whale sharks. The Watershark navigates the water collecting plastics and microplastics, alien vegetation, and floating debris.

The task of underwater litter collection is currently primarily performed by human divers. In addition, the EU projects MAELSTROM [82] and Natural Seabed [192] are developing large cable robots[1] to pick up underwater litter, reducing dependence on heavy and expensive marine support. However, Natural Seabed's robotic system is not autonomous, and both solutions focus on large cable robots; therefore, they are not very agile and adaptable. Hence, these systems are constrained when employed in complex topographies, such as ports in which marine debris tends to accumulate, and where large vessels and platforms cannot easily enter or be installed. Furthermore, a prototype for an underwater litter collection robot has been developed within the Rozalia Project [265]. This robot is operated remotely by humans. Finally, in [283] a jellyfish-inspired robotic platform is presented, integrating electro-hydraulic actuators and a hybrid structure of rigid and soft components. The robot can perform a wide range of functions for diverse applications, including litter collection. This is not an integrated approach for cleaning up the sea though, as this 16 cm-diameter robot can only collect a small volume of litter, and it does not involve a stand-alone large-sized collection basket for depositing the litter.

Contrary to these solutions, the SeaClear system introduces an autonomous, integrated approach to detect and collect marine litter, featuring an adaptable architecture. To the best of our knowledge, SeaClear is the first system that

[1] Cable robots are a type of robotic manipulators where the end-effector is manipulated by multiple cables controlled by winches.

consolidates all these advanced functionalities into a single solution. This emphasis on integration, adaptability, and automation significantly enhances the current state-of-the-art in marine litter collection technologies.

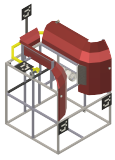
4.3. SEACLEAR SYSTEM DESCRIPTION

The SeaClear architecture (Figure 4.1) has been developed with the aim to search for and collect marine waste in an autonomous way. The system consists of five main components: an Unmanned Surface Vehicle (USV) that scans the sea bottom, a collection basket for storing the litter, an observation Remotely Operated Vehicle (ROV) that searches for litter underwater, a collection ROV that collects the litter, and an Unmanned Aerial Vehicle (UAV), which searches for litter from the air.

Although some hardware components like sensory equipment and the aerial drone are standardized products available on the market, other parts such as the underwater and surface vehicles, the gripper, the collection basket, as well as the respective launch and recovery systems are specifically developed towards the SeaClear purposes and tailored to fulfill the task of underwater litter clean-up efficiently. This section describes each robotic component comprising the SeaClear system. The tables with the technical details can be found in Appendix 4.B.



(a) SeaCat USV.



(b) Collection basket.



(c) Observation (mini Tortuga) ROV.



(d) Collection (Tortuga) ROV.



(e) Aerial drone UAV.

Figure 4.2: SeaClear system modules.

4.3.1. ROBOTIC MODULES

UNMANNED SURFACE VEHICLE - SEACAT USV

The SeaCat USV (Figure 4.2a) is a sea-going unmanned surface vehicle originally developed for offshore wind farm inspections. It is 6.83 m in length, 3.1 m in width, and 2.15 m in height when inflated. The SeaCat USV features a range of

sensors, such as a high-resolution multibeam echosounder, a Differential Global Positioning System (GPS), a Real-Time Kinematics (RTK) GPS, and sensor data interfaces including serial, USB, and Ethernet ports, facilitating various connectivity options with the rest robotic vehicles. It serves as a central platform, offering the capability to deploy not only the other three robotic vehicles, but also a collection basket for placing the litter. Each component is tethered to the SeaCat and released by an individual Launch and Recovery System (LARS). Importantly, the SeaCat is equipped with a high-resolution multibeam echosounder. This enables it to perform bathymetry scans of the seafloor, creating accurate 3D-models of the seabed. Large underwater litter may already be identified through this initial survey.

Collection basket

The collection basket (Figure 4.2b) is tethered to the SeaCat USV and is lowered to the seafloor with an individual winch from the rear part of the USV. Once on the seafloor, the basket also serves as an anchor to the SeaCat USV, while the two underwater ROVs perform their tasks in detecting and collecting litter. The basket itself is designed to hold up to approximately 250 kg of litter per clean-up operation and is made from aluminium struts to provide sufficient stability. These struts are connected via a net with approximately 2 cm large meshes to present less risk of accidentally entrapping wildlife. Due to its funnel-shaped entrance and its bristled interface, it allows precise alignment of the collection ROV and a safe deposit of litter without the risk of it floating out of the basket again. The localization of the basket underwater is achieved through fusing data using a Kalman Filter from the included sensors on the basket such as the Inertial Measurement Unit (IMU), the depth sensor, and the Short Baseline (SBL) localization sensor. In addition, the basket is also equipped with lights and ArUco markers for a more precise positioning of the collection ROV during the close-range maneuvering. An integrated camera inside the basket enables the operator to observe the litter deposit process, assess the litter intake capacity, and identify potential problems.

Observation ROV - mini Tortuga ROV

The Observation (mini Tortuga) ROV, depicted in Figure 4.2c, is a lightweight inspection robotic vehicle that employs several sensors for underwater debris detection. This sensing system is aimed to distinguish among debris, fauna, and flora, ensuring that only debris is collected without harming any living organism. It combines the use of a conventional full-HD video camera with high-power LED lights to survey the surroundings primarily in clear or low-turbidity water, and an integrated dual-frequency multibeam imaging sonar to augment visibility under poor light conditions. The task of the Observation ROV is to create a map of the detected litter from these sensors. During this task, the previously generated map from the SeaCat USV is further refined. The Observation ROV is connected to the SeaCat USV via a tether, more specifically a power line communication cable. The winch for the Observation ROV is located at the bow of the SeaCat USV, where the launch and recovery are performed.



(a) Individual modules in an unloaded state.



(b) Modules when loaded on the SeaCat USV.

Figure 4.3: SeaClear system components.

COLLECTION ROV - TORTUGA ROV

The Collection (Tortuga) ROV is a larger (inspection class) ROV designed to operate in strong currents and to collect debris from the seafloor thanks to an embedded gripper, as can be seen in Figure 4.2d. Similarly to the Observation ROV, the Collection ROV also features a computer-vision-based system. The Collection ROV locates debris through an integrated downward-facing camera in the gripper, which provides images of the trash to be picked up and guides the robot to the right spot for grasping it. The operation of the Collection ROV shares similarities with its smaller counterpart, the Observation ROV. The Collection ROV is also tethered to the SeaCat USV by a winch, which is situated at the rear of the SeaCat USV.

The Collection ROV collects the waste and transfers it to the collection basket. The collection mechanism consists of a gripping and a suction tool that are capable of collecting various types of litter. The suction device is aimed to assist the gripper during debris collection for diverse object types, such as plastic wrappers or bags.

Two critical considerations that have been taken into account during the design phase of the collection mechanism, include: (1) the requirement for a skid to

contain the gripper and the suction device and to be mechanically connected to the Collection ROV, and (2) the need that all components minimize their negative hydrodynamic influence to retain good controllability of the Collection ROV in water. Given these considerations, the gripping and the suction tools have been designed such that they consist of two shovels for litter collection and a pump with a hose that can draw objects toward the suction cup. The gripper, featuring a streamlined and bio-inspired design, was developed and manufactured using a metal 3D printing process. This allowed to create a mesh-like structure from aluminum-magnesium alloys that achieves a balance between strength and weight, as well as allow any small animals that were accidentally trapped to escape.

The gripper and the suction device can be operated independently of each other through the Collection ROV's communication interface. The two shovel-like segments of the gripper are driven by a single motor controlled by a motor driver that is connected to the Collection ROV. The motor and its driver are contained inside a casing on which the gripping segments are mounted. The inputs to the gripper are generated by the automatic controller, although it is also possible to provide manual input.

Aerial drone - DJI Matrice 210 RTK V2 UAV

The UAV surveys the area in which the robotic platform operates and enables detection and mapping of locations with larger litter concentrations that are visible from the air [255]. This provides initial information for a more detailed search performed by the underwater robots in the next step. The detected litter is placed in the map used to guide the Observation ROV so as to perform close-up scans. The UAV is depicted in Figure 4.2e.

In conditions where the waters are transparent, the UAV can also contribute to the localization of the ROV. This can be particularly useful given the fact that the underwater robots work in a GPS-denied environment and hence the UAV serves as an alternative source for the position information of the ROV. In this way, when the ROV is detected near the surface, it gets supplied with a GPS estimate based on the drone's RTK-GPS and the localization in its camera frame, improving the ROV's localization accuracy.

Following the initial free-flight survey, the UAV is connected to the SeaCat USV through a tether. The 40 m long tether enables continuous air operation without the need to land by supplying power and maintaining a reliable communication channel. At the same time, it imposes a constraint on the movement of the UAV, necessitating the coordination with the USV. The UAV LARS is the key component in this coordination task. Situated high at the SeaCat USV bow, it consists of a landing platform, a cable winch, and associated electronics. Hence, it functions as the power and communication interface between the aerial and surface vehicle.

4.4. Software Architecture for System Integration

The software architecture of the SeaClear system comprises two main components: the Shore Operation Centre (SOC) and the Robotic System (Figure 4.4). The SOC

contains a collection of server-sided applications that host a web-based application accessible to registered users. Because it provides services that facilitate users to monitor and interact with the Robotic System, the web-based application is also called the SeaClear Service Layer. The SOC also serves as the access point for professional human intervention and provides high-level commands to the Robotic System. It enables direct commanding of the Robotic System and processing of user requests, by means of planning a mission, setting a target position, or enabling and disabling the controllers. It ensures a point-to-point communication with the Robotic System and, at the same time, it exchanges mission information with the web-based application. The SeaCat USV, the two ROVs, and the UAV are all connected through a Robot Operating System (ROS) framework. The Robotic System is connected to the SOC via wireless communication, predominantly using 5 GHz WiFi. Sensory data is exchanged directly through native ROS applications, or, in some cases, through ROS-bridge TCP or websockets. Visual perception is enhanced through 1080 p video streams that are directly fed from the onboard cameras through an RTSP[2] protocol.

A digital twin of the Robotic System is connected within the aforementioned architecture similar to the real system, enabling rapid development and offering a cost-effective testing environment for control and navigation of the SeaClear Robotic System. Multiple SeaClear systems, real or digital-counterparts, could be integrated within the same SeaClear Service Layer, provided that the correct transport layer settings are made within each Local Area Network (LAN) where the fleet of robots operates.

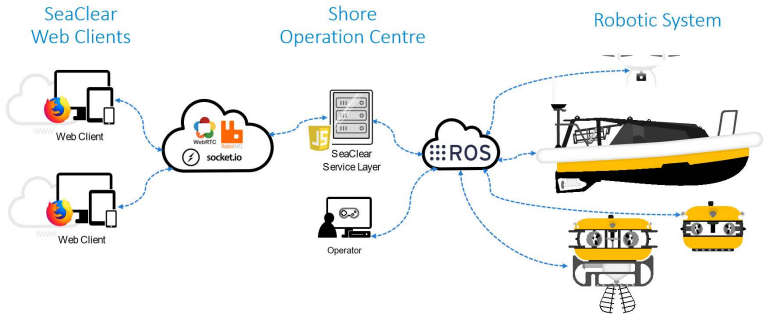


Figure 4.4: Overview of the SeaClear software architecture.

4.5. PROPOSED SYSTEM WORKFLOW & METHODOLOGY

4.5.1. OVERVIEW

The mode of operation for the SeaClear system is depicted in Figure 4.5. When the system reaches the designated area for its mission, the SeaCat USV establishes an initial bathymetry scan of the sea floor. Large pieces of litter may already be detected in this step. If the water is clear enough, the aerial drone searches from

[2]https://datatracker.ietf.org/doc/html/rfc2326

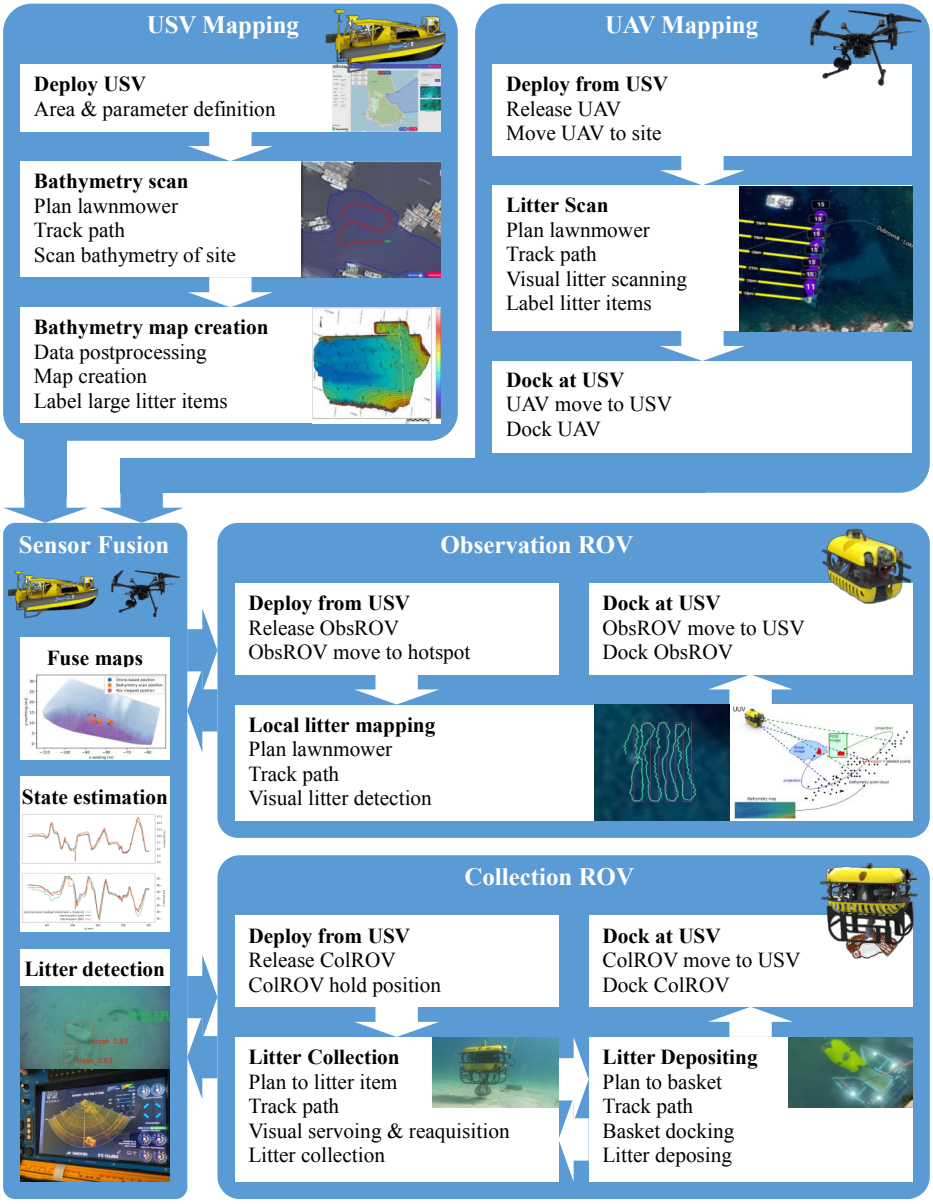


Figure 4.5: Overview of the SeaClear mode of operation.

above. The information of the USV and the aerial drone is then refined into a map, which is sent to the Observation ROV, equipped with the sonar and video imaging systems. The images captured by the Observation ROV are analyzed using object detection algorithms to differentiate between live organisms and litter fractions. The latter are marked in the initial map, which is constantly refined as the Observation ROV follows a lawnmower trajectory while searching the area. After that, the Collection ROV is deployed, navigating to the litter items marked on the map. It then collects the detected litter and deposits it in the collection basket. The litter that cannot be collected and deposited into the collection basket, i.e. due to its large size, remains marked in the map for collection with suitable equipment later on. Once the collection process is finished, all vehicles tethered to the SeaCat USV are autonomously recovered from the water and the SeaClear robotic platform returns to the shore.

4.5.2. POSE ESTIMATION

The pose (i.e. the position and the orientation) of the ROVs is the most challenging to estimate, since they travel underwater. We rely on sensor fusion between their IMUs, Doppler Velocity Loggers (DVLs) mounted on their undersides, and an absolute position sensor. For the latter, underwater acoustic sensors are expensive and sometimes unavailable, so as an alternative we find the global ROV position using the camera image of the UAV [53]. This is of course only possible when the ROV is close enough to the surface to be detected in the UAV image stream, so an interesting intermittent-measurement estimation problem arises [171, 177]. We first describe the positioning algorithm in the UAV image, followed by the overall pose estimator that fuses all the available sensors.

An object tracking algorithm is executed on the UAV camera image to output the pixel position of the ROV, which is then used to obtain the normalized camera-referenced coordinates ${}^{C}\overline{\mathbf{X}}$, assuming calibrated intrinsic parameters of the camera. Further along the position estimation pipeline (Figure 4.6) the UAV altitude (a_{UAV}) and the ROV depth (d_{ROV}) are combined to obtain the distance to ROV operating plane. Define

$$\mathbf{p}_0 = (0, 0, a_{\text{UAV}} + d_{\text{ROV}})^{\text{T}} \tag{4.1}$$

$$\mathbf{n} = (0, 0, 1)^{\text{T}} \tag{4.2}$$

By using camera extrinsic matrix ${}^{C}R \in SO(3)$, obtained from the angles reported by the camera gimbal, allows us to recover the distance d of the ROV to the camera's optical center:

$$d = \frac{{}^{C}\mathbf{p}_0 \cdot {}^{C}\mathbf{n}}{{}^{C}\overline{\mathbf{X}} \cdot {}^{C}n} \tag{4.3}$$

where ${}^{C}\mathbf{p}_0 = {}^{C}R\,\mathbf{p}_0$ and ${}^{C}\mathbf{n} = {}^{C}R\,\mathbf{n}$.

Metric coordinates of the camera-referenced ROV position ${}^{C}\mathbf{P}$ are then obtained by inverse projection of the normalized coordinates:

$${}^{C}\mathbf{P} = d\,{}^{C}\overline{\mathbf{X}}. \tag{4.4}$$

Finally, the GPS position and georeferenced orientation of the drone are used in order to transform the camera-referenced position of the ROV into a global position estimate. For a more comprehensive description and details of transformations between coordinates, readers are encouraged to refer to the original paper [53].

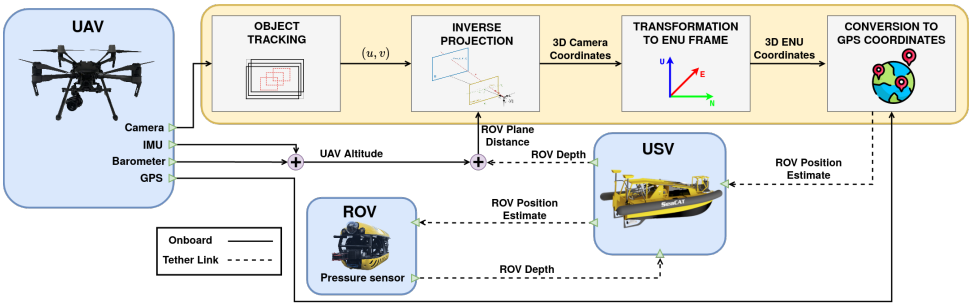


Figure 4.6: Scheme of the ROV position estimation from the UAV adapted from [53]. The ROV reports its own current depth, necessary to compute the ROV position estimate onboard the UAV. This estimate is then reported back to the ROV and used as an external input for the full ROV pose estimation algorithm.

To fuse all the sensors (onboard and, when available, UAV-based) the ROV performs pose estimation using an extended Kalman filter. The detailed implementation is described in [187], and uses a kinematic model with 6 degrees of freedom for the robot. The equations for the model and the extended Kalman filter updates are standard, so they will not be given here, with one exception: due to the intermittent nature of the UAV-based position measurement, the measurement equation of the system switches between two possibilities. It is given by

$$\mathbf{y}_k = C_k \mathbf{x} + v_k \tag{4.5}$$

where the output matrix C_k takes one of two values: C_o or C_f. Option C_o corresponds to on-board sensors only, so measurements include angular position and velocity from the IMU, linear velocity from the DVL, and vertical position from a water pressure sensor. Option C_f is applied at those time steps k where the UAV-based position is available. It adds two extra rows to C_o, corresponding to the position in the plane from the UAV. The output and noise dimensions increase accordingly. The performance of the estimator is evaluated later on, in Section 4.6.

4.5.3. AERIAL AND UNDERWATER MAPPING

AERIAL DEBRIS DETECTION

Detecting submerged marine debris from UAVs poses significant challenges stemming from various factors. The efficient absorption of light by water causes rapid attenuation with depth. This issue of limited visibility is further exacerbated by surface wave motion, resulting in motion blur and distorted images. Additionally,

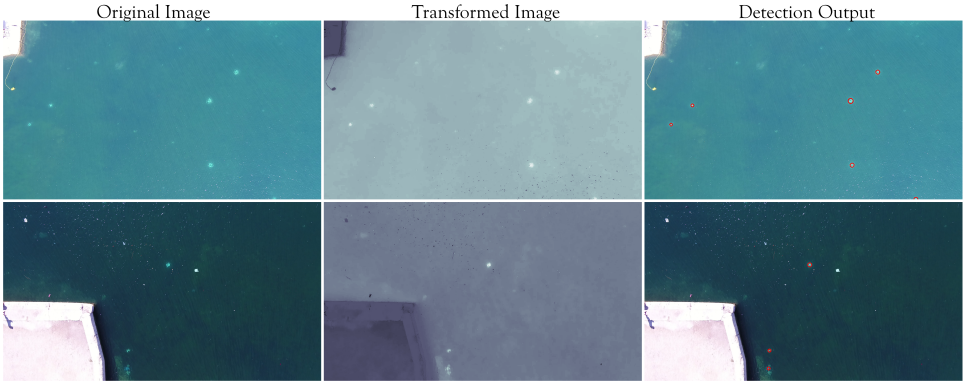


Figure 4.7: Visualization of the inputs and outputs of the aerial marine debris detection pipeline. From left to right, the figure shows the captured input images, the transformed images utilized as an input for the blob detection phase, and the final detection outputs visualized on the original images.

the considerable distance between the UAV and the submerged objects often results in captured images lacking details, with objects being represented merely as small blobs. Moreover, the phenomenon of glare, caused by reflections at the water-air interface, introduces specular highlights in images, potentially leading to erroneous detection of non-existent objects.

Since there are no publicly available datasets that would facilitate learning-based methods for underwater object detection from aerial cameras, we utilize a learning-free approach based on a Difference of Gaussians (DoG) blob detector. We handcraft the inputs of the blob detection method based on experimentally observed image statistics by taking the exponential of the difference between the blue/green color channel and the red color channel:

$$I_{b-r} = \exp({}^{b}I - {}^{r}I), \tag{4.6}$$

$$I_{g-r} = \exp({}^{g}I - {}^{r}I), \tag{4.7}$$

where ${}^{c}I$ denotes the intensity of the color channel $c \in \{r, g, b\}$ of the image I.

This allows us to produce the blob detector input as a transformed single channel image $I_{\text{input}} = \frac{(I_{b-r}+I_{g-r})}{2}$ which can be interpreted as score map where high values indicate the possible presence of a submerged object. Intuitively, this approach based on the difference of color channels utilizes the fact that for submerged objects wavelengths in the red range are attenuated rapidly. As seen in the transformed images shown in Figure 4.7, applying this transformation helps to eliminate surface objects and specular highlights (typically saturated in all color channels of the original image) which could interfere with the blob detection algorithm further along the pipeline.

Blob detection can be formulated as the task of finding the local extrema in the image scale-space [163]. The scale-space representation of an image is obtained by

the convolution of an image $I(x, y)$ with a variable-scale Gaussian kernel $G(x, y, s)$:

$$G(x, y, s) = \frac{1}{2\pi s} \exp \frac{x^2 + y^2}{2s}$$

The scale-space can thus be defined by the following function:

$$L(x, y; s) = I(x, y) * G(x, y, s) \tag{4.8}$$

where $*$ represents the convolution operator.

For a scale-space representation obtained at a specific scale s applying the Laplacian operator $\nabla^2 L = L_{xx} + L_{yy}$ results in a strong response for blobs of radius $r^2 = 2s$. Typically multi-scale detection is achieved by simultaneously considering local extrema with respect to both space and scale by using the scale-normalized Laplacian:

$$\nabla^2_{\text{norm}} L(x, y; s) = s\nabla^2 L(x, y; s) = s(L_{xx} + L_{yy}). \tag{4.9}$$

The DoG operator [175] is an approximation of the Laplacian operator, generalizing to the scale-normalized Laplacian case with scale levels given by $\sigma_{i+1} = k\sigma_i$:

$$\text{DoG}(x, y; s) \approx \frac{(k^2 - 1)}{2} s\nabla^2 L(x, y; s) = \frac{(k^2 - 1)}{2} \nabla^2_{\text{norm}} L(x, y; s) \tag{4.10}$$

We utilize the *scikit-image* implementation of the DoG blob detector, which requires the user to define the σ_{min} and σ_{max} hyperparameters related to the minimum and maximum blob size, respectively. Since σ is given in pixel units we define minimum and maximum radii r of objects of interest in meters and use the following heuristic function to output σ_{min} and σ_{max} based on the UAV height h measurements and known intrinsic camera parameters (DFOV, image size):

$$\sigma(r, h) = r \frac{\sqrt{I^2_{\text{width}} + I^2_{\text{height}}}}{2h \tan \frac{\text{DFOV}}{2}} \tag{4.11}$$

$$\sigma_{\text{min}} = \sigma(r_{\text{min}}, h) \tag{4.12}$$

$$\sigma_{\text{max}} = \sigma(r_{\text{max}}, h) \tag{4.13}$$

Examples of the detection output, based on two samples of experimental data collected in Marseille, France, can be observed in Figure 4.7.

Deep-learning-based, underwater debris detection

The underwater detection of debris to further refine the map is realized by the Observation ROV. During this task, objects are detected, localized, and classified as debris, plants, animals, or ROV parts respectively, using a Convolutional Neural Network (CNN) algorithm, selected for its capability to extract hierarchical features from images, which is necessary for the detection and classification of objects in underwater environments [115]. Various CNN-based approaches exist for the task of object detection, ranging from faster region-based CNNs (R-CNNs) to YOLO

networks [222, 223]. Faster R-CNNs can achieve high-performance object detection with pixel-level accuracy. However, the inference time for this type of CNNs is not sufficiently fast to support real-time application when processing the video input from the ROV's camera. In addition, R-CNN requirements in terms of memory and storage impose limitations when they are implemented on robotic hardware, due to the lack of large-scale cooling mechanisms in these applications, which are necessary in order to support operation under such an intensive load. Hence, YOLO is selected, since the SeaClear system is intended for real-time application with the aim to achieve not only a high accuracy but also a fast inference speed.

To properly train the YOLO neural network, a large custom dataset is generated by data collected during different data collection trials. The synthesis of a sufficiently large and good-quality dataset is not trivial, considering that it should contain observations taken in varying conditions, to ensure the generalization capability of the trained neural network. This also requires accurate ground-truth annotations. Due to the scarcity of publicly available datasets for underwater debris, it was necessary to generate our own dataset. The dataset consists of 8610 labeled image samples, containing observations of various types of debris, animals, plants, and ROV parts [51]. The annotations are in the form of bounding boxes around the target objects.

The YOLO architecture (YOLOV6 S, 17.2 M parameters) is trained and tested on this generated dataset. The training was done based on the implementation provided in the official repository of this architecture[3]. The YOLO network was trained for 200 epochs with a batch size of 32 and an input image size of 640×640.

The performance metrics of object detection, such as Precision (P), Recall (R), and the F1 score, are necessary for evaluating the effectiveness of the YOLO network [81]. Precision measures the accuracy of the predictions, indicating the proportion of positive detections that were actually correct, while Recall assesses the ability of the neural network to find all relevant instances in the dataset. The F1 score is the harmonic mean of Precision and Recall and it provides a single score that balances both metrics.

In addition to these metrics, the mean Average Precision (mAP) and the Average Precision at various Intersection over Union (IoU) thresholds are important in evaluating the performance of object detection techniques. The IoU criterion measures the overlap between the predicted bounding box and the ground truth, with a higher IoU indicating a more accurate prediction. The Average Precision corresponds to the area under the precision-recall curve, and it provides a summary measure that combines the precision and recall into a single metric. The mAP is the mean of the average precision values across all classes and IoU thresholds, providing a single score to represent the overall precision and recall of the neural network across different object categories [98].

The results of training the YOLO network are presented in Table 4.1. The YOLO network achieves a performance equal to 93.5% mAP at 0.5 IoU threshold, while specifically for the trash, the YOLO network can achieve a performance of 92.9% mAP at 0.5 IoU threshold. It should be noted that, as anticipated, the mAP exhibited

[3] https://github.com/meituan/YOLOv6

a decline at higher IoU thresholds. This is consistent with the nature of the mAP@0.5:0.95 metric, which averages the mAP over a range of IoU thresholds from 0.5 to 0.95, thereby affecting the true positive classification rate. In addition, the implementation of image enhancement methods was also explored, but they did not yield a significant increase in performance and hence they were not incorporated into the real-time application [118].

Class	Labeled images	Labels	P@.5iou	R@.5iou	F1@.5iou	mAP@.5	mAP@.5:.95
all	1722	6321	0.925	0.89	0.907	0.935	0.711
rov	172	204	0.975	0.94	0.957	0.974	0.876
plant	84	92	0.932	0.89	0.91	0.92	0.649
trash	1542	3333	0.932	0.89	0.91	0.929	0.725
animal	685	2692	0.915	0.85	0.881	0.918	0.594

Table 4.1: Evaluation of YOLO performance.

To improve the accuracy and reliability of the YOLO-based detection module, a strict confidence threshold mechanism is implemented in combination with a temporal validation method. Each detected object in YOLO is associated with a confidence score, ranging from 0 to 1, representing the neural network's certainty that the bounding box accurately includes a specific class of object. We have implemented a high confidence threshold of 0.85 in our approach. This ensures that only detections with a high likelihood of correctness are considered, thereby significantly reducing the probability of false positives. In addition to the confidence threshold, our approach incorporates a temporal validation strategy to address the challenge of repetitive or erroneous detections in successive frames. This approach involves a time-based criterion for detection validity, where a detection must persist for a certain duration, between 2 and 3 seconds, to be considered valid. This duration is chosen based on empirical observations, ensuring a balance between detection responsiveness and accuracy. In this way, we filter out transient detections that are likely to be false positives.

The image frames from the video stream of the ROV's camera are fed as input to the YOLO neural network, which then outputs, for each detected object, the class label, the confidence score, and the coordinates of the bounding box around the object (Figure 4.8).

Litter mapping

Once the objects have been detected, they must be placed on a map for later pickup by the Collection ROV. The SeaClear pipeline for doing this is shown in Figure 4.9. Litter detected in the UAV camera stream is registered in a 2D occupancy grid. The ROV-based map is maintained as a point cloud, initialized from the bathymetry scan performed by the SeaCat USV. Litter detected in the Observation ROV sonar and camera streams is registered by labelling points in this point-cloud based, reference map.

Methodologically, we perform a forward projection for detections by both the camera and the sonar on the Observation ROV, using a separate projection model



Figure 4.8: Example result of the YOLO object detection algorithm.

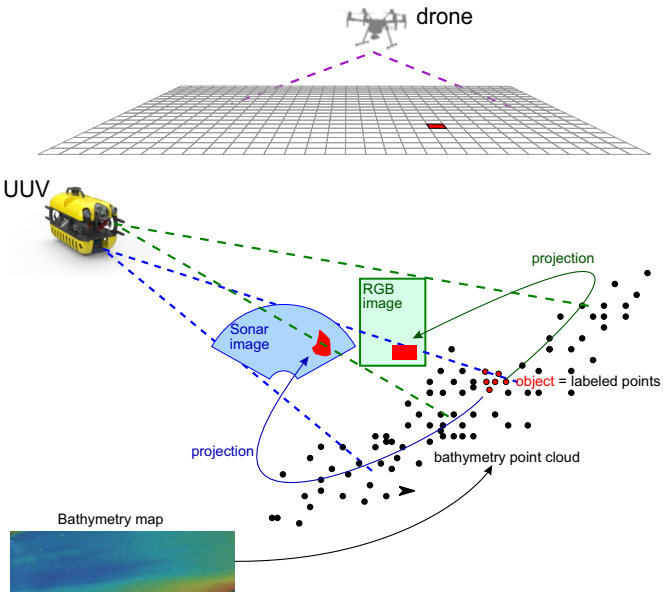


Figure 4.9: Mapping framework.

for each sensor. The projection methods for the camera and sonar are illustrated in Figure 4.10, left and right respectively. A pinhole camera model is used for which the parameters are obtained by performing a calibration in seawater. The image coordinates p are obtained from the homogeneous world coordinates of the point P_w using

$$p = KTP_w \tag{4.14}$$

where T is the transform between the camera and the world coordinate frames, and K is the intrinsic matrix. For the sonar, the model assumes that a point in the

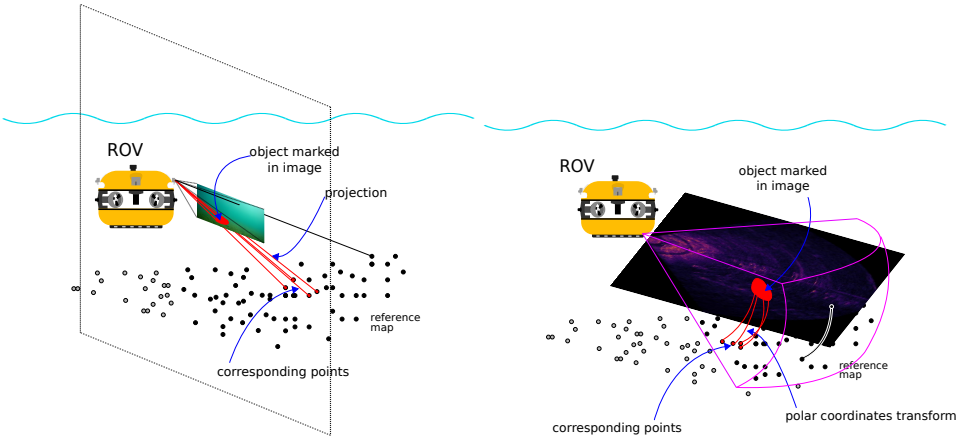


Figure 4.10: **Left:** Mapping from a marked camera image. The pre-selected points are shown in black, and the projection from the point cloud into the image by lines. The red object in the camera image corresponds via the red lines to the object points in the point cloud, also shown in red. **Right:** Mapping from a marked sonar image. The symbolism is similar to the camera mapping. One difference is that here the black points are pre-selected using the more complicated shape of the sonar field of view, shown in a red outline.

volume visible by the sensor produces an image in the zero-elevation image plane at the range and azimuth of the point. The image frame produced by the sonar, in this case, is a finite, scaled, and discretised subset of the zero elevation image plane. Thus, the projected point p in the image plane can be found using the polar coordinates of this point, consisting of range R and azimuth angle θ:

$$p = R \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{4.15}$$

For both projection methods, we compute where a pre-selected part of the bathymetry point cloud would fall in the image given the ROV pose, and subsequently, points that intersect the mask of the litter detected in the image are labeled. Point pre-selection is performed to reduce computational requirements. For the camera, the half-plane in front of the ROV is pre-selected, while for the sonar we preselect points found within the bounding azimuth and elevation angles reported by the sonar itself. These points define a cone-like shape, illustrated in magenta in Figure 4.10, right. A mechanism to prevent duplicate detections from subsequent image frames is also implemented: when a litter item is detected closer than a threshold to an existing item, instead of adding a new item, a first-order filter is applied to move the existing item towards the new detection. The threshold is tuned based on the precision of the pose estimate.

A final task in mapping is to design the ROV trajectory. A simple lawnmower

is applied in the field experiments, but in simulations and proof-of-concept experiments in the lab we are also investigating methods to find more efficient paths, based either on reinforcement learning [226] or model-based planning [304].

4.5.4. LITTER COLLECTION

Once objects have been cataloged in the reference map, they are ready for collection by the Collection ROV.

PATH PLANNING

To reach a litter location safely, various path planning frameworks have been considered [69, 289]. A collision-free path is planned using informed-RRT* [69], aided by a local Nonlinear Model Predictive Control (NMPC) planner with cascaded (position and velocity) control (see Figure 4.11).

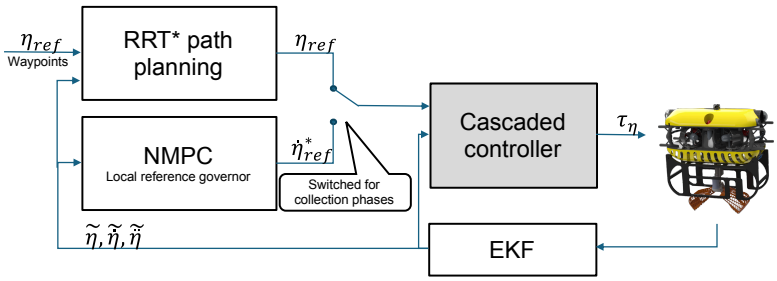


Figure 4.11: Control structure of the Collection ROV with switching reference inputs during path tracking and the collection sequence.

The planner works directly in the ROS and it is based on the octomap[4] that is generated during the bathymetry scan and mapping process. An octomap is based on octrees, an efficient hierarchical data structure for spatial subdivision, and uses probabilistic occupancy estimation for representing volumetric 3D environment models [112]. An example from our simulation can be found in Figure 4.15 in Section 4.6.1.

VISION-BASED COLLECTION STRATEGY

For collecting litter objects with the Collection ROV, we define a strategy with three steps after reaching the estimated litter location: *searching, visual-servo tracking* and *collection*, as illustrated in Figure 4.12.

Searching It should be noted that both the pose estimation during the litter mapping and the pose estimation of the Collection ROV show estimation errors (up to 1.3 m Root Mean Square Error (RMSE), see Section 4.6.2), which cumulatively can lead to the Collection ROV missing the true litter location and having the object

[4] https://octomap.github.io

outside the field of view of the downward-facing camera. To reacquire litter location, a spiral search pattern is tracked in the xy-plane with fixed height and constant velocity.

Visual-servo tracking Upon detecting a litter object, we employ a vision-based Nonlinear Model Predictive Controller (NMPC) based on [101] for collection with the Collection ROV. Using the frame definitions shown in Figure 4.12, the goal is to coincide the body-fixed frames of the gripper $C_{\mathcal{G}}$ with the object of interest $C_{\mathcal{O}}$, tracking the object visually in the camera frame $C_{\mathcal{C}}$ (with a static transformation to $C_{\mathcal{G}}$). This robust sensor-based control approach ensures accurate object grasping and (since it works in a local camera reference frame) does not depend on the pose accuracy in world coordinates. However, it introduces additional challenges to maintain the object within the camera's field of view (visibility constraint), as well as constraining the velocity of the ROV. A more detailed description is provided in the upcoming Section 4.5.4.

Collection Close to the object persistent visibility[5] may not be guaranteed anymore. The algorithm then switches to an open-loop grasping phase based on the last recorded pose of the litter target. Hence, we employ the extended Kalman filter described in Section 4.5.2 to fuse data from DVL, IMU, Attitude and Heading Reference System (AHRS), and downward-looking camera for robust state estimation with respect to the relative motion to the estimated target pose. Grasping is initiated based on an altitude threshold determined by the DVL with respect to the object and seafloor.

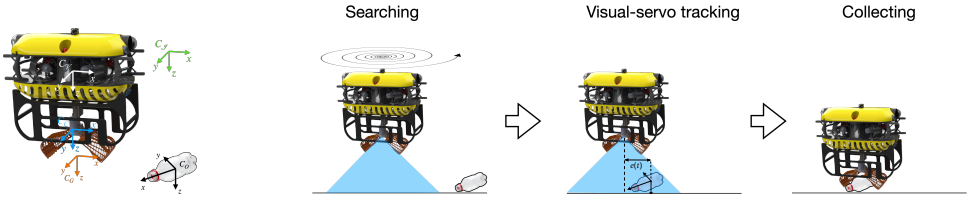


Figure 4.12: (Left) Depiction of the inertial frame ($C_{\mathcal{I}}$), the vehicle-frame ($C_{\mathcal{V}}$), the body-fixed frames of the camera ($C_{\mathcal{C}}$), the gripper ($C_{\mathcal{G}}$), and the frame of the object of interest ($C_{\mathcal{O}}$) (Right) Grasping phases: If no object is directly found at the map coordinates, a spiral search pattern is followed until a litter object is detected in the gripper-camera image. With the object in sight, visual-servo tracking centers the object in the gripper frame during the decent. Once the object is in reach it is collected for transportation to the basket.

[5] determined by the object dimensions and the field of view of the camera

NONLINEAR MODEL PREDICTIVE CONTROL

Here we provide details to the vision-NMPC controller to guide the robot to the collection position while respecting operational limitations defined below. Consider an underwater vehicle with dynamics as described in [62], equipped with a complete AHRS scheme, a gripper, a camera system and an online object detection algorithm. Given the real-time pose of the camera relative to the object frame provided by the vision system, ${}^{\mathcal{O}}\boldsymbol{\eta}_{\mathcal{C}}(t)$, and the fixed, known pose of the camera frame $C_{\mathcal{C}}$ with respect to the gripper frame $C_{\mathcal{G}}$, the pose of the gripper frame relative to the object frame ${}^{\mathcal{O}}\boldsymbol{\eta}_{\mathcal{G}}(t) = \left[{}^{\mathcal{O}}\boldsymbol{p}_{\mathcal{G}}^{T}(t)\ {}^{\mathcal{O}}\boldsymbol{\phi}_{\mathcal{G}}^{T}(t)\right]^{T}$ can be readily computed. Considering the collection point as the desired pose of the gripper with respect to the object frame i.e., ${}^{\mathcal{O}}\boldsymbol{\eta}_{\mathcal{G}}^{d}$, we could define the error between the gripper and collection point as

$$\boldsymbol{e}(t) = {}^{\mathcal{O}}\boldsymbol{\eta}_{\mathcal{G}}(t) - {}^{\mathcal{O}}\boldsymbol{\eta}_{\mathcal{G}}^{d}(t) \tag{4.16}$$

In sampled data NMPC, a finite-horizon optimal control problem is solved at discrete sampling time instants t_j based on the current measurements $\boldsymbol{e}(t_j)$, with prediction horizon T_P and a constant sampling time $0 < h < T_P$, such that $t_{j+1} = t_j + h,\ \forall j \geq 0$. Thus, the control signal applied in between the sampling instants is given by the solution of the following finite-horizon optimal control problem:

$$\min_{\hat{\boldsymbol{\nu}}_{\mathcal{V}}(\cdot)} J(\boldsymbol{e}(t_j), \hat{\boldsymbol{\nu}}_{\mathcal{V}}(\cdot)) = \tag{4.17a}$$

$$\min_{\hat{\boldsymbol{\nu}}_{\mathcal{V}}(\cdot)} \left\{ \int_{t_j}^{t_j+T_p} F\Big(\hat{\boldsymbol{e}}(\tau), \hat{\boldsymbol{\nu}}_{\mathcal{V}}(\tau)\Big) d\tau + E\Big(\hat{\boldsymbol{e}}(t_j + T_P)\Big) \right\}$$

subject to:

$$\text{System dynamics [62]: } \dot{\hat{\boldsymbol{e}}} = f(\hat{\boldsymbol{e}}(t), \hat{\boldsymbol{\nu}}_{\mathcal{V}}(t)) \ , \tag{4.17b}$$

$$\hat{\boldsymbol{e}}(t_j) = \boldsymbol{e}(t_j), \qquad \hat{\boldsymbol{\nu}}_{\mathcal{V}}(t_j) = \boldsymbol{\nu}_{\mathcal{V}}(t_j), \tag{4.17c}$$

$$\hat{\boldsymbol{\nu}}_{\mathcal{V}}(t) \in V_{\mathcal{V}}, \qquad {}^{\mathcal{O}}\boldsymbol{\eta}_{\mathcal{C}}(t) \in H_{\mathcal{C}} \tag{4.17d}$$

$$\hat{\boldsymbol{e}}(t_j + T_P) \in \mathcal{E}_f \tag{4.17e}$$

where $\mathcal{E}_f$ is a terminal region around the desired configuration and F and E are the running and terminal cost functions respectively which are both of quadratic form i.e., $F(\cdot) = \hat{\boldsymbol{e}}^{\top}\boldsymbol{Q}\hat{\boldsymbol{e}} + \hat{\boldsymbol{\nu}}_{\mathcal{V}}^{\top}\boldsymbol{R}\hat{\boldsymbol{\nu}}_{\mathcal{V}}$ and $E(\cdot) = \hat{\boldsymbol{e}}^{\top}\boldsymbol{P}\hat{\boldsymbol{e}}$, respectively, with $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ being positive definite matrices to be appropriately tuned [101].

Field of view constraints $H_{\mathcal{C}}$ The underwater robot should be able to always keep the object of interest within the camera field of view during the visual-servo tracking operation. Given the dimensions of the object[6] and the camera's horizontal and vertical angles of view α_h and α_v as shown in Figure 4.13, a set of inequalities can be defined that must be satisfied for this. These inequalities are the 3D extended version of the field of view constraints presented in our previous work [100]. A detailed description of the inequalities is provided in Appendix 4.A.

[6] Real-time bounding-box along the horizontal and vertical axes coming from the algorithm described in Section 4.5.3.

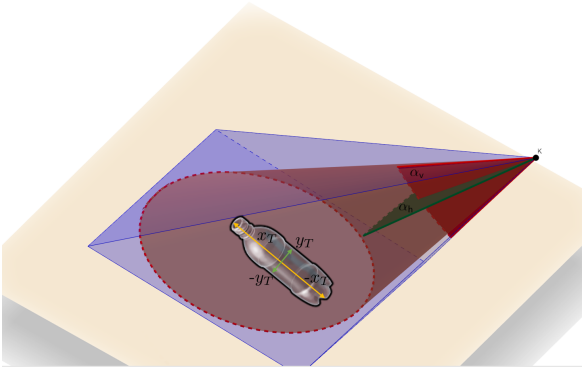


Figure 4.13: Modeling of the camera visibility limitations for an object with dimensions y_T and z_T. The camera's horizontal and vertical angles of view are denoted by α_h and α_v respectively. The inner cone describes a set of soft constraints to keep the object within the field of view, whereas the pyramid forms the set of hard constraints of the camera's field of view.

Velocity constraints $V_\mathcal{V}$ In order to allow the vision tracking system to identify the object more frequently [70, 228], it is required to impose a prescribed smoothness to the system motion. In particular, the vehicle is required to move relatively slowly with upper bound velocity vector $\bar{\boldsymbol{\nu}}_{\mathcal{V}_1} = [\bar{u}_\mathcal{V},\ \bar{v}_\mathcal{V},\ \bar{w}_\mathcal{V}]^\top \in V_\mathcal{V} \subset \mathbb{R}^3$, where $\bar{(\cdot)}$ denotes the corresponding upper bounds for each coefficient and $V_\mathcal{V}$ is the constraint set given by:

$$V_\mathcal{V} := \left\{ \boldsymbol{\nu}_\mathcal{V} \in \mathbb{R}^3 : |u_\mathcal{V}| \leq \bar{u}_\mathcal{V}, |v_\mathcal{V}| \leq \bar{v}_\mathcal{V}, |w_\mathcal{V}| \leq \bar{w}_\mathcal{V} \right\} \tag{4.18}$$

4.5.5. COORDINATION BETWEEN UAV AND USV DURING OPERATION

Throughout the underwater mapping and collection phase, the UAV is tethered to the SeaCat USV, hovering above the survey area in order to provide position feedback for the ROVs and to enhance situational awareness. During this operation, the UAV needs to maintain the position with respect to the SeaCat USV, due to the kinematic constraint imposed by the tether. This control task is split between the UAV and the LARS.

The role of the UAV is to control the relative position with respect to the SeaCat USV. To estimate the relative position, an extended Kalman filter is implemented based on continuous RTK measurements obtained from the two vehicles. Moreover, when in sight, the estimator is provided with the intermittent relative position information obtained from the ArUco markers located on the landing platform. This

additional measurement is necessary to obtain centimeter-level landing accuracy imposed by exceptionally small landing platform area ($1\,m^2$). Given the relative position estimate, a proportional controller is employed for set-point tracking, with its output feeding into the UAV's proprietary velocity controller.

To prevent potential entanglements and excessive pulling, especially during the take-off and landing phase, the tether needs to be continuously winched in and out to match the UAV motion. This is achieved by controlling the UAV LARS motor velocity. The deployed cable length is estimated based on the LARS motor encoder data. Then, using the L_2 norm of the estimated UAV position as the length reference signal, a Proportional-Integral-Derivative control loop runs on the LARS onboard computer and outputs the motor velocity reference. This velocity reference is subsequently passed to the integrated LARS motor driver. The overall tethered UAV control scheme is shown in Figure 4.14.

4.6. EXPERIMENTAL RESULTS AND ANALYSIS

4.6.1. SIMULATION MODEL

To enable quick initial testing during the design phase and remote development without a real marine environment, we have used robot simulation tools such as Gazebo. We developed our own simulation environment based on `UUVsimulator`[7] [179] for the underwater part, `Virtual RobotX`[8] (VRX) [18] for the water-level part, and `RotorS`[9] [67] for the aerial domain. These elements are extensions of Gazebo, a software tool for simulating physical systems such as robots, vehicles, and environments. Gazebo supports realistic sensors, kinematics, and environments via various plugins. We chose Gazebo because of its high modularity through *plug-in*s and its strong connection to ROS, which we use for our control pipeline. This allowed us to test and prototype the software pipeline without the expense of real-world testing, leaving the fine-tuning for real-life marine testing occasions. The creation of digital twins of both the robot systems and the test sites helped to lower the hurdles for transferring the algorithms to the real system and enabled the conceptual functionality test of the full system in an environmental setting mimicking the demo sites. An illustration of such a functionality test for the collection pipeline is shown in Figure 4.19.

Test site digital twins were created via a fusion of bathymetry data (sonar scans) for the seabed model, and Google Maps 3D data for the surface models. For litter mapping and collection tasks, the digital twins can be randomly populated with litter models. The resulting overall model is imported as a Collada (.dae) file into the Gazebo simulation environment and used for visualization as well as a collision mesh for objects and sensors.

Visualizations of the terrain for the Petroleumhafen (petrol harbor) in Hamburg, Germany, can be seen in Figure 4.16a, and the combined render with a simulated water volume in Figure 4.16b.

[7] https://github.com/uuvsimulator/
[8] https://github.com/osrf/vrx
[9] https://github.com/ethz-asl/rotors_simulator

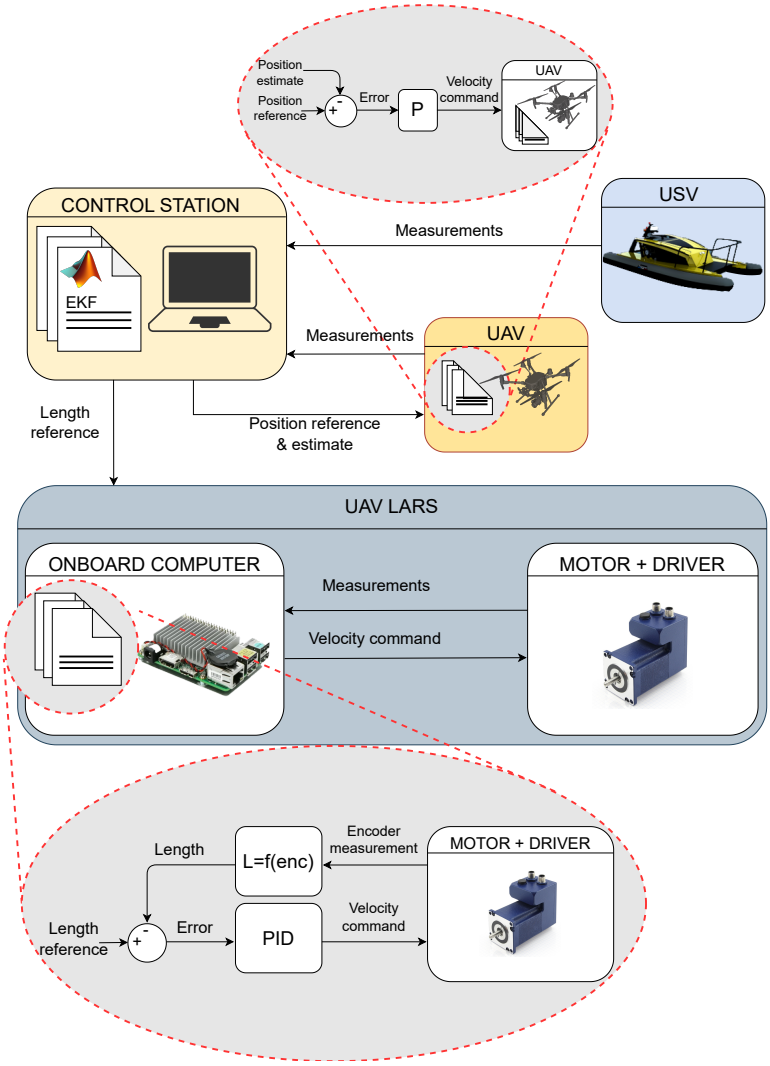


Figure 4.14: Tethered UAV control scheme overview.

The robot digital twins are created via a fusion of CAD data and kinematics of the robot and an estimate of its drag using linear and quadratic approximations. Integrating a robot's geometry and actuator locations into the SeaClear simulator is very similar to an existing framework in Gazebo ROS. The description of the robot utilizes the Unified Robot Description Format (URDF) with Xacro.

Sensors are added onto a simulated vehicle by using Gazebo plugins, mostly readily available from `UUVsimulator`. The cable has a significant influence on ROV dynamics, and therefore, a cable model has been developed to obtain a more

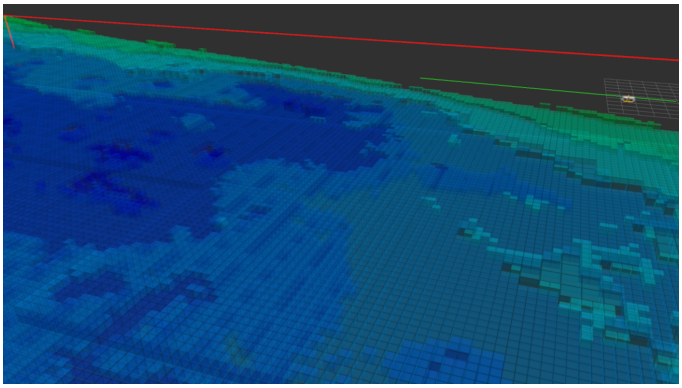
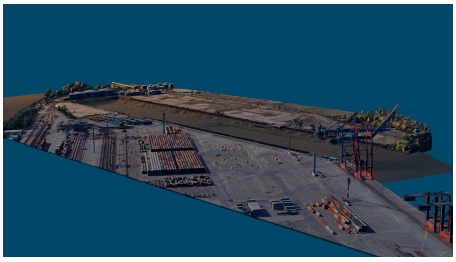
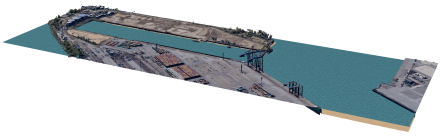


Figure 4.15: Red: The planner bounds. Green: A downsampled path planned by the RRT* algorithm in the simulated octomap created from real bathymetry data.



(a) 3D terrain render



(b) Gazebo water-bound render

Figure 4.16: Hamburg port as a digital twin.

realistic system model for the digital twin. The beam-based tether dynamics and simulation use a finite element model for modeling the movement of the umbilical in 3D and is successfully implemented in simulation for a large number of links. Further details are presented in [27]. Figures 4.17a to 4.18b show visualizations of the digital twins.

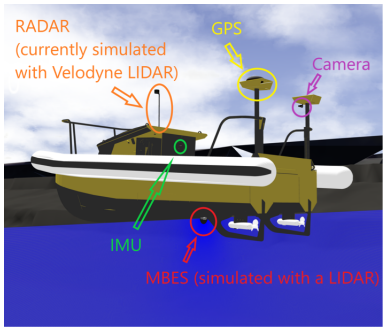


(a) Collection (Tortuga) ROV.



(b) Observation (mini Tortuga) ROV.

Figure 4.17: ROV digital twins in Gazebo.



(a) SeaCat USV and sensor integration.



(b) Simulation of the Tethered DJI Matrice drone.

Figure 4.18: Rest digital twins in Gazebo.

To showcase the simulation, Figure 4.19 displays a simulated sequence of the automated litter collection with the Collection ROV. For the sake of clarity in presentation, after deployment the simulation time between 10 s and 38 s has been cut from the plot to allow the system to settle to the initial conditions before receiving its first command. The robot follows a series of waypoints until it receives a litter location from mapping at (-1.3m, 0.2m). Note that this does not coincide with the true litter location at (-2.1m, 0.9m), so that the object is outside the field of view of the camera. Therefore, the search strategy for re-aquisition of the litter item is followed until the downward-facing camera detects the litter object for at least 2 seconds. As long as the object is detected, the visual-servo tracking then ensures movement of the Collection ROV to bring the object to the image center and align with the gripper. At the required minimum altitude over the object, determined by the DVL, the gripper is activated to grasp the litter object.

4.6.2. FIELD EXPERIMENTS

Multiple field experiments were carried out in real-world settings in Hamburg, Germany, and in Dubrovnik, Croatia (specifically Lokrum island and Bistrina Bay) to confirm the effectiveness and applicability of the system that was designed. In this section, we present some results concerning the ROV pose estimation, control, mapping, object detection, and collection obtained during these field experiments.

FIELD EXPERIMENTS INFRASTRUCTURE

The SeaClear operations have been planned taking in consideration a minimal infrastructure for deploying the robotic system. Specifically, for the Hamburg port environment a 20 ft container placed on a floating pontoon has been used as land station, with a direct sight towards the harbor basin. The wireless equipment has been mounted on the roof of the container, ensuring at least 5 m height above

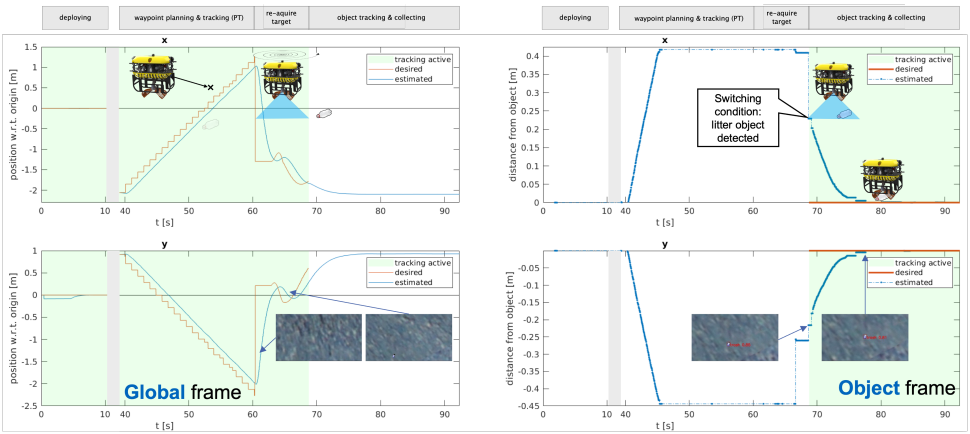


Figure 4.19: Simulation of the full automated collection sequence based on live object detection. The green shaded areas highlight the active tracking in global and object coordinate frame, respectively. Upper plots show the x-coordinate, lower plots the $y-$coordinate in the respective frame. Both plots show convergence of tracking in the object frame towards the center of the camera frame for picking up the object with the camera-aligned gripper.

sea level. In the test site around Lokrum island, the equipment was placed on the lower deck of the carrier ship, at a height of about 4 m above sea level. The antenna used was a 5G Hz, 19/20 dbi, 120 deg sector antenna, which is guaranteed by the manufacturer for a few kilometers, with a reasonable drop of bandwidth. The electromagnetic waves are susceptible to various environmental conditions, including temperature, humidity, precipitation and are impacted by the reflections from the water surface. In addition, the wireless communication link was limited to a single Point-To-Point connection from the USV to shore, having a frequency and channels tuned by the manufacturer to be hidden to the other wireless network clients and therefore minimising the noise on the wireless channel. On land, the sector antenna was connected to the operational team via cable. During demonstrations, the number of clients to high-bandwidth data, such as video streams was limited to the operator monitoring the specific process. Moreover, high-payload data, such as sonar images, uncompressed images, or point cloud data have been omitted from the operational log (in the form of rosbags) and saved on the corresponding companion computers. A summary of the bandwidth allocation and the acceptable latency is provided in Table 4.2.

LITTER DETECTION

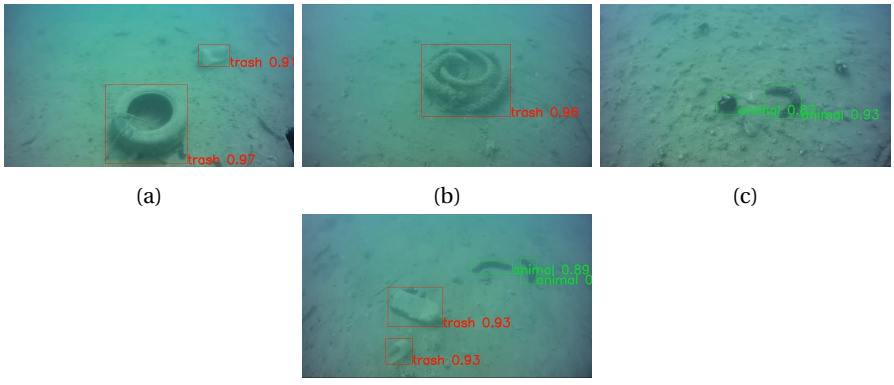
Figure 4.20 presents a series of detections captured during the field test in Bistrina, Dubrovnik, Croatia. The YOLO algorithm demonstrated an 87% accuracy rate in litter detection. This validates the system's capability to effectively identify various

Data Type	Frequency	Bandwidth Allocation [Mbit]	Latency [ms]
Text	1 Hz-1 kHz	0.8	40-60
PointCloud	300 000 pps	20	1000
RTSP HD VideoStream (Operator)	30 fps	50	150
Web HD VideoStream (Web client)	30 fps	50	3000

Table 4.2: Wireless communication system bandwidth allocation and acceptable latencies (tested within LAN).

types of marine debris. Specifically, the system recognized both larger objects and smaller items, such as rubber tires, plastic bottles, and entangled ropes, which are commonly found in underwater environments. In addition to its litter detection capabilities, the algorithm also demonstrated a high level of accuracy in recognizing marine life. This functionality to differentiate between litter and marine fauna highlights the algorithm's performance at the critical feature of preventing ecological harm during the cleanup process.

However, it is important to acknowledge the algorithm's limitations in adapting to varied environmental conditions. The tests near Lokrum island, conducted in shallow waters approximately 15 m deep, revealed that changing lighting conditions significantly affected the algorithm's performance. This limitation was particularly evident when the environmental conditions deviated from those of the training dataset, indicating a need for further development in this area.



(a) (b) (c)

(d)

Figure 4.20: Object detection results during the field experiment.

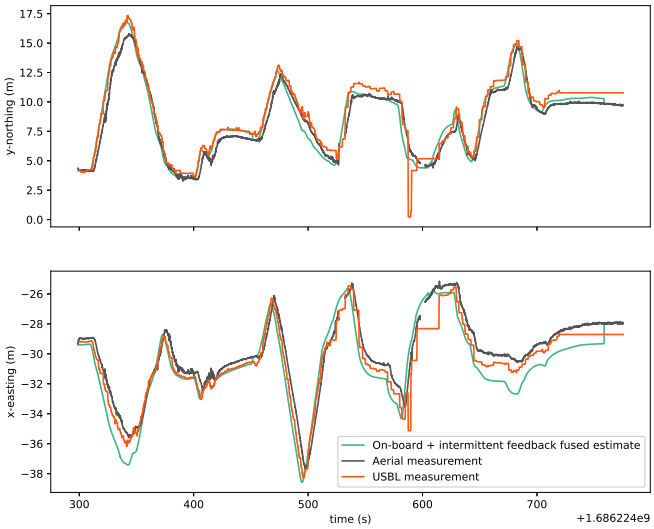


Figure 4.21: Pose estimation result: evolution over time of the planar positions X and Y. The horizontal axis depicts Unix epoch time in seconds.

ROV POSE ESTIMATION

Figure 4.21 shows the estimated planar position of the ROV in an experiment run in Hamburg, Germany. The UAV-based position is shown in black, and this is fed back into the ROV pose estimator with a period of 153 s, where it is fused with onboard sensors, and the resulting estimate is shown in green in the figure. To investigate the accuracy of this estimate, we compare it against SBL acoustic positioning, shown in orange, and the RMSE obtained is 1.3 m.

Figure 4.22 shows the influence of the period with which the UAV-based position is read on the RMSE compared to the SBL reference. As expected, errors generally increase with larger periods.

ROV CONTROL

A series of waypoints describing a lawnmower trajectory was defined for the Observation ROV to follow so as to perform mapping of a litter hotspot area detected in Dubrovnik, Croatia. Figure 4.23 shows these waypoints as small disks and the real trajectory of the ROV. Note that the waypoints are also connected by lines to clarify things visually, but in reality the ROV runs a setpoint regulator that aims to reach the next waypoint, rather than trajectory tracking. The curved trajectories followed in-between waypoints are due to the fact that the robot is strifing laterally, and unmodeled tether dynamics and dynamical couplings are affecting the control;

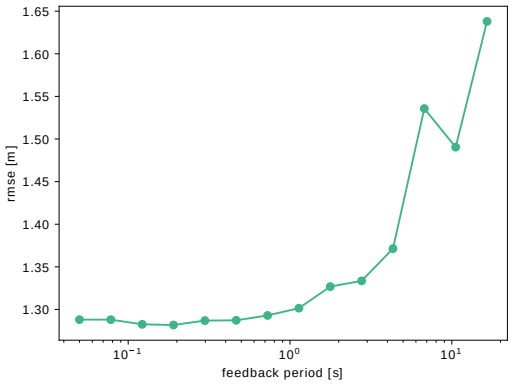


Figure 4.22: RMSE between pose estimate and SBL obtained for a range of UAV feedback period choices.

nevertheless they are useful to cover more area with the sensors.

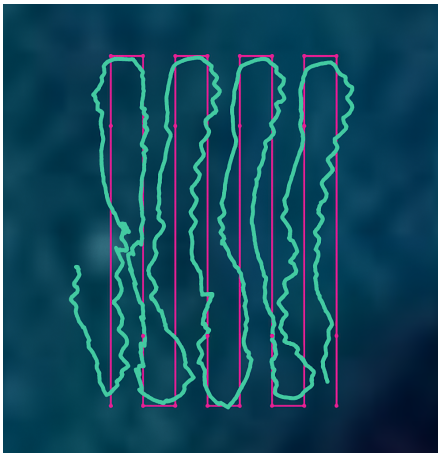
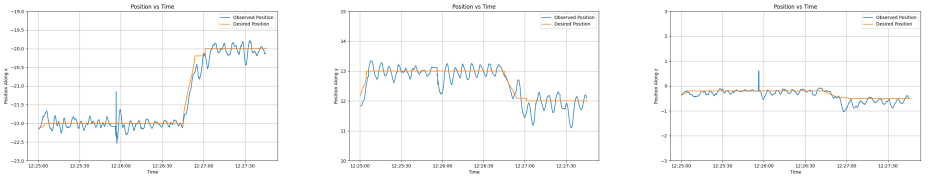
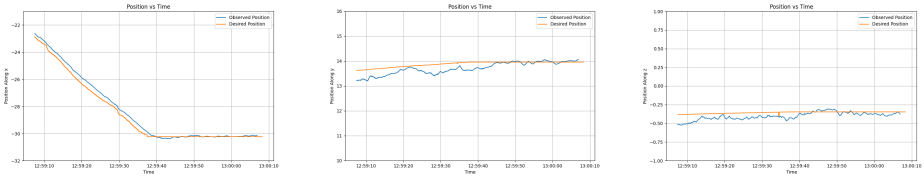


Figure 4.23: Planned waypoints (magenta disks) connected by lines, and actual trajectory (teal) of the ROV.

During SeaClear demonstrations at the Hamburg test site, data was collected from position tracking during experiments with the Collection ROV. Raw data was collected from sensory, state estimation, and control input data to track errors in both position and velocity. Figures 4.24a and 4.24b show the desired positional references and observed positions via odometry from an extended Kalman filter, respectively. The observed error consistently remains below 0.5 m and the RMSE values for positions consistently stay below 0.3 m.



(a) Track and station-hold



(b) Long range tracking

Figure 4.24: Tracking performance of the Collection ROV from Hamburg port demonstrations.

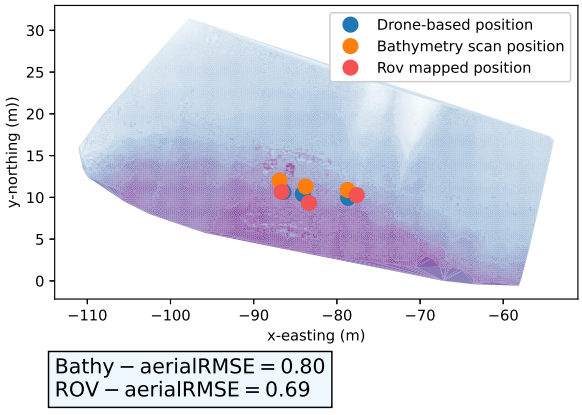


Figure 4.25: Mapped litter positions.

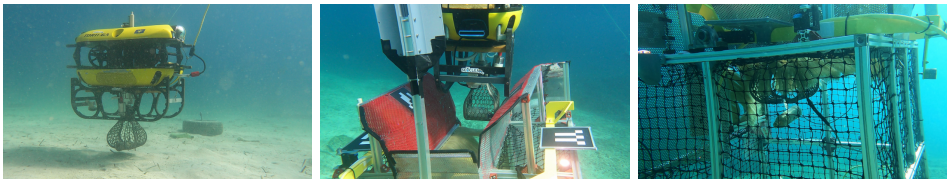
LITTER MAPPING

Figure 4.25 shows the mapping results in an experiment run in the same location as the pose estimation experiment above. The bathymetry point cloud is plotted, on top of which we show the positions of three identified litter items. Three versions of each position are shown: as manually identified on the bathymetry scan, as detected in the UAV image, and finally, as automatically mapped in the sonar image through the ROV mapping pipeline described earlier. The UAV detection serves as a means

to validate the others; note that in general, finding underwater objects on the UAV image will not be possible. All three sets of positions are in good agreement with each other, with the ROV detections being slightly closer to the UAV ones.

Litter collection

Figure 4.26 shows the collection and deposition of litter during the field test in Lokrum, Dubrovnik, Croatia, in October 2023.



(a) Litter collection (b) Basket entry (c) Litter deposition

Figure 4.26: Litter collection sequence during field trials at the Lokrum, Dubrovnik, Croatia testsite.

Tethered UAV control

The effectiveness of the proposed control structure for the tethered UAV was showcased during the final SeaClear demonstrations in Dubrovnik, Croatia. The UAV LARS system was deployed from the support vessel *Naše more*. Figure 4.27 shows the trace of the tethered UAV GPS position across all flight stages, encompassing take-off, set-point tracking, and landing. The system demonstrated its capability to autonomously execute the entire flight task, therefore validating the UAV control approach. A snippet from the UAV onboard camera during the experiment is given in Figure 4.28.

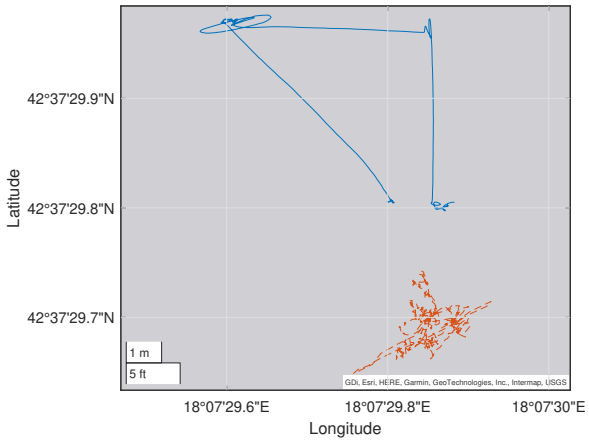


Figure 4.27: UAV GPS position (blue) and UAV LARS GPS position (red dashed) in the SeaClear Lokrum Demo.



Figure 4.28: A view from the tethered UAV onboard camera during the SeaClear Lokrum Demo. The UAV LARS is located in the center of the image.

4.7. CONCLUSIONS AND FUTURE WORK

This chapter has provided a comprehensive description and analysis of the SeaClear system, the first autonomous multi-robot system to collect litter from the seabed using advanced control and AI methods. This description includes a detailed explanation of the system's infrastructure, the function of each module, and the design strategy for the sensing and control scheme of the multi-robot platform. Field tests in Hamburg, Germany, and near Dubrovnik, Croatia have demonstrated autonomous detection, mapping, and collection capabilities, showcasing the system's performance as well as the potential of robotics technology to assist or reduce human involvement, enhance operational efficiency, and minimize the risk associated with marine litter collection.

As lessons learned, it is critical to highlight the inherent difficulties encountered in accurate sensing within underwater environments. Precise estimation of the position and orientation of the ROVs pose a non-trivial challenge, yet these metrics are essential for tasks like mapping, path planning and picking up small objects. Moreover, the efficacy of object detection is heavily reliant on sensor quality, while underwater conditions, such as water properties and lighting, often differ significantly from those in the training set, creating a discrepancy between the algorithm's training environment and its operational setting. In addition, persistency of detection over a longer timespan with shifting points of view and lighting conditions is much more challenging than object detection in single image frames, but crucial for the continuous tracking during visual-servo control. Finally, it should be emphasized that significant challenges arise in the transition from simulation to real-world testing in validating the architecture due to the constrained and often simplified environment and hardware representations.

Future work will focus on extending the current computer vision architecture and integrating it with additional state-of-the-art computer vision algorithms. Moreover, further analysis is necessary to enhance the interpretability of the neural networks in use, which is critical for validating their decisions in underwater environments [237]. Developing effective data augmentation strategies is also crucial to address the variability encountered in different underwater environments.

A few important next steps to improve mapping are: better pose estimation by using better sensors and algorithms; using octtree litter representations instead of point clouds, together with Bayesian fusion from cameras and sonars; and developing a self-localization and mapping (SLAM) pipeline. For precise control of the Collection ROV during the grasping operation, we plan on including online learning-based control strategies that account for uncertainties in the environment, such as currents, as well as unknown dynamics of the system, e.g. mass and drag changes from picking up litter. Furthermore, we plan to increase the capabilities of the system for lifting larger and heavier items. In addition, we intend to enhance the robustness of ROV tracking from the UAV through the implementation of tracking failure detection mechanisms.

Ultimately, the paramount goal is to develop a comprehensive strategy to tackle the complete life cycle of marine litter. This can be achieved not only by scaling up the current SeaClear architecture to multiple teams of autonomous robots, but also

by empowering community involvement and taking steps to raise public awareness about marine litter. We hope that this research will pave the way for future investigation and advancements in the field, contributing to maintaining cleaner seas.

Appendix

4.A. Field of view requirements

Let us first define vectors ${}^{\mathcal{O}}\boldsymbol{p}_c = [{}^{\mathcal{O}}x_c, {}^{\mathcal{O}}y_c, {}^{\mathcal{O}}z_c]^\top$ and ${}^{\mathcal{O}}\boldsymbol{\phi}_c = [{}^{\mathcal{O}}\phi_c, {}^{\mathcal{O}}\theta_c, {}^{\mathcal{O}}\psi_c]^\top$ describing the pose of the camera frame C_c with respect to the object frame $C_{\mathcal{O}}$, and α_h and α_v the field of view angles in horizontal and vertical axes (see Figure 4.13).

Considering y_T and z_T to be the dimensions of the object with respect to the object frame, the following inequalities describe the field of view constraints of the systems, that must be satisfied in order to always maintain the object within the camera field of view. These requirements are captured by the constraint set $H_{\mathcal{C}}$:

$$ {}^{\mathcal{O}}\boldsymbol{\eta}_{\mathcal{C}} = \left[{}^{\mathcal{O}}\boldsymbol{p}_{\mathcal{C}}^T \; {}^{\mathcal{O}}\boldsymbol{\phi}_{\mathcal{C}}^T \right]^T \in H_{\mathcal{C}} \tag{4.19} $$

which can be defined as

$$ -{}^{\mathcal{O}}y_c + {}^{\mathcal{O}}z_c \tan\left(-{}^{\mathcal{O}}\phi_c - \frac{\alpha_h}{2}\right) - \frac{y_T}{2} \geq 0 \tag{4.20a} $$

$$ {}^{\mathcal{O}}y_c - {}^{\mathcal{O}}z_c \tan\left(-{}^{\mathcal{O}}\phi_c + \frac{\alpha_h}{2}\right) - \frac{y_T}{2} \geq 0 \tag{4.20b} $$

$$ {}^{\mathcal{O}}x_c + {}^{\mathcal{O}}z_c \tan\left({}^{\mathcal{O}}\theta_c - \frac{\alpha_v}{2}\right) - \frac{z_T}{2} \geq 0 \tag{4.20c} $$

$$ -{}^{\mathcal{O}}x_c - {}^{\mathcal{O}}z_c \tan\left({}^{\mathcal{O}}\theta_c + \frac{\alpha_v}{2}\right) - \frac{z_T}{2} \geq 0 \tag{4.20d} $$

For locally grasping the identified object of interest, we consider that the vision tracking algorithm provides in real-time the pose of the camera with respect to the local inertial frame $\mathcal{I}$ i.e., ${}^{\mathcal{I}}\boldsymbol{p}_c$ and ${}^{\mathcal{I}}\boldsymbol{\phi}_c$ with $C_{\mathcal{I}} \approx C_{\mathcal{O}}$, as well as an estimation of the object dimensions, i.e., y_T and z_T as a bounding box along the horizontal and vertical axes on the object frame. In the SeaClear system, this is fulfilled by the object detection algorithm updating the bounding box and center of geometry with 30 fps.

4.B. Technical specifications

Type	Catamaran, aluminum hull, inflatable lateral tubes
Dimensions	L: 6.83 m, W: 3.1 m, H: 2.15 m (inflated) L: 5.78 m, W: 2.15 m, H: 2.15 m (deflated)
Draught	0.73 m (empty), 0.9 m with max payload
Weight	1075 kg without payload
Payload	500 kg
Max. speed	6 knots
Max sea state	4 (operation), 6 (transit)
Power supply	Generator, 8 kW diesel engine
Tank capacity	2 x 200 L for up to 8 days autonomy
Propulsion system	2 unidirectional Aziprop electrical thrusters
Mode of operation	Manual or auto navigation
Navigation sensors	High-resolution multibeam echosounder, 2 full HD video cameras, DGPS (RTK in option), Gyrocompass, Radar, AIS, INS
Communication	2.5 GHz WiFi antennas with 5 km range, 2 back-up UHF antennas (900/1200 MHz and 430 MHz), 4 underwater GPS receivers
Sensor data interfaces	Serial, USB, Ethernet

Table 4.3: Technical specifications of the SeaCat USV

Type	Inspection class ROV
Dimensions	L: 672 mm, W: 310 mm, H: 381 m
Weight	20 kg in air, neutral in water
Payload	5 kg (wet weight) and 10 kg (dry weight)
Max. speed	4 knots
Power supply	220 Vac (main power or generator)
Propulsion system	4 horizontal thrusters (3 kgf per thruster) with azimuthal adjustment and 2 vertical thrusters (3 kgf per thruster)
Mode of operation	Manual or auto navigation
Sensors	Full-HD video front camera pan & tilt (1920x1080 p), LED lights (2x 10 000 lumen), compass, pressure (depth) sensor, temperature, leak detector, AHRS (inertial navigation system), multibeam sonar Oculus 750d, USBL, altimeter, DVL Teledyne Wayfinder, WaterLinked A50
Tether	500 m, 9.5 mm neutral / 8 mm negative buoyant, breaking strength 500 kg
Communication	Power line communication (PLC)
Sensor data interfaces	Serial, USB, Ethernet

Table 4.4: Technical specifications of the observation (mini Tortuga) ROV

Type	Inspection class ROV
Dimensions	L: 996 mm, W: 430 mm, H: 461 mm (without gripper)
Weight	40 kg in air, neutral in water (without gripper)
Payload	5 kg (wet weight) and 10 kg (dry weight)
Max. speed	4 knots
Power supply	220 Vac (main power or generator)
Propulsion system	4 horizontal thrusters (3 kgf per thruster) with azimuthal adjustment and 2 vertical thrusters (3 kgf per thruster)
Mode of operation	Manual or auto navigation
Sensors	Full-HD video front camera pan & tilt (1920x1080 p), rear camera pan & tilt, compass, pressure (depth) sensor, temperature, leak detector, AHRS (inertial navigation system), multibeam sonar, USBL, altimeter, DVL
Auto functions	Heading, depth, altitude
Tether	500 m, 9.5 mm neutral / 8 mm negative buoyant, breaking strength 500 kg
Communication	Power line communication (PLC)
Sensor data interfaces	Serial, USB, Ethernet

Table 4.5: Technical specifications of the collection (Tortuga) ROV

Dimensions (unfolded)	L: 883 mm, W: 886 mm, H: 427 mm
Weight	4.69 kg
Payload	1.23 kg
Max. speed	Ascent: 5 m/s Descent: 3 m/s
Max. wind resistance	12 m/s
Flight time	No limit due to umbilical power supply
Integrated sensors	Dual barometer and dual IMU, upward infrared sensor, first-person view (FPV) forward vision system, aircraft status indicator, antennas (relay aircraft control and video signals)
Hovering accuracy (GPS)	Vertical: ± 0.5 m Horizontal: ± 1.5 m
Hovering accuracy (D-RTK2)	Vertical ± 0.1 m Horizontal: ± 0.1 m
Tether length	40 m
Camera	Zenmuse X5S (3-axis stabilization)

Table 4.6: DJI Matrice 210 RTK V2 UAV

5

Towards Establishing an Automated Selection Framework for Underwater Image Enhancement Methods

The majority of computer vision architectures are developed based on the assumption of the availability of good-quality data. However, this is a particularly hard requirement to achieve in underwater conditions. To address this limitation, plenty of underwater image enhancement methods have received considerable attention during the last decades, but due to the lack of a commonly accepted framework to systematically evaluate them and to determine the likely optimal one for a given image, their adoption in practice is hindered, since it is not clear which one can achieve the best results. In this chapter, we propose a standardized selection framework to evaluate the quality of an underwater image and to estimate the most suitable image enhancement technique based on its impact on the image classification performance.

This chapter is based on the paper: A. Ilioudi, B. J. Wolf, A. Dabiri, and B. De Schutter. "Towards establishing an automated selection framework for underwater image enhancement methods". In: *OCEANS 2023 - Limerick*. 2023, pp. 1–6. DOI: 10.1109/OCEANSLimerick52467.2023.10244710.

Author contributions statement: A. Ilioudi conceptualized the proposed architecture, developed the code for the architecture implementation, conducted the related experiments, and wrote the manuscript.

5.1. INTRODUCTION

State-of-the-art image processing methods are gaining traction rapidly for underwater applications. Advanced techniques are emerging from the latest achievements in the field of computer vision. Enabled by the latest computer vision advancements, novel, sophisticated, underwater image processing methods have been developed, allowing underwater robotic vehicles to perceive their surrounding environment in order to achieve particularly complex tasks, such as litter detection, equipment inspection, fish monitoring, and tracking of ecosystems. Yet, an interesting paradox hovers over the development of underwater image processing techniques: the more approaches emerge, the less straightforward it is to determine which one is the most suitable to be implemented in each case.

More specifically, the fast pace of computer vision breakthroughs has accelerated the spread of image processing techniques in underwater applications in order to compensate for the degraded visibility conditions in the water. In general, the quality of underwater images is particularly low, due to various distortions caused by absorption, scattering, and other characteristics that affect the vision under the water. Due to the water, which is a particularly complex and inhomogeneous medium and also due to absorption and scattering of light, less light can enter a camera during image captioning, in comparison with an environment without water. An example to illustrate the degraded underwater visibility conditions is to consider that knowing the true color of an object underwater is almost not possible even for humans, since in these conditions, human vision is distorted as well.

Underwater image enhancement is a key technology in compensating for the poor visibility conditions and enabling underwater robots to perform complex tasks autonomously. However, the degradation variations among different water environments hinder the process of selecting the enhancement technique that is most suitable to be implemented in each case [152]. Furthermore, the currently existing underwater Image Quality Assessment (IQA) metrics, especially the ones that do not take into account a reference image (no-reference), have not yet been validated with respect to their relation to the performance of computer vision tasks. In other words, IQA metrics do not evaluate an enhancement method on the basis of the performance increase they yield for image classification, which is a fundamental objective of employing image enhancement approaches.

The aim of this work is to present a standardized analysis of the impact of image enhancement techniques on underwater images as well as to propose an automated framework that evaluates the quality of an underwater image and estimates the best image enhancement technique based on the impact of each technique on the image classification performance. The potential advantages of this framework are also discussed. Our analysis is complementary to the existing works in the field of underwater image enhancement. The key differences are that (1) our focus is different and is primarily on integrating the impact on the image classification performance, (2) we introduce an automated selection approach for categorizing image enhancement techniques based on the image classification performance, and (3) we provide valuable insights about the image enhancement effect during pre-processing and post-processing of the training data. Specifically, this work

includes five main steps:

- We implement three image enhancement techniques on raw underwater image data namely Fusion [4] as well as the neural networks WaterNET [154], and Cast-GAN [151], and we generate an image dataset for each implemented technique.
- We evaluate a Convolutional Neural Network (CNN) that performs image classification on each one of the above-mentioned generated datasets [295].
- We propose an image quality assessment method based on the image classification performance of the trained CNN.
- We analyze the quality of an underwater image in terms of comparing the performance of human-based image classification, algorithmic-based image classification, as well as, in terms of the respective quality rated by underwater image quality metrics.
- We train an additional supervised classification model that relates images, their IQA metrics, and the CNN's performance with image enhancement techniques.

The chapter is organized as follows: Section 5.2 presents an overview of necessary preliminary knowledge. In Section 5.3 the proposed scheme is presented and analyzed. In Section 5.4 the implementation of the proposed architecture is presented. Section 5.5 demonstrates the results and Section 5.6 gives concluding remarks and an outlook to future work.

5.2. PRELIMINARIES

This section presents some background and material with regards to evaluation metrics and methods that are analyzed within this study.

5.2.1. IMAGE QUALITY ASSESSMENT METRICS

Performance validation and comparison among underwater image enhancement methods remains a largely unexplored research area. As an alternative to commonly used subjective tests, IQA metrics are implemented as objective measures to quantify perceptual quality [152]. Underwater IQA metrics measure color and contrast degradation. Then they quantify the perceived image quality via image attributes related to the degradation in water and they combine image attributes to mimic human preferences in the enhanced images. IQA metrics that are commonly used to assess the quality of enhanced underwater images are Underwater Image Quality Measure (UIQM) [200], Underwater Color Image Quality Evaluation (UCIQE) [300], and Colorfulness, Contrast, and Fog density index (CCF) [285]. However, it has been reported that underwater-specific IQA measures do not satisfactorily rate the quality of enhanced underwater images as it is not clear whether a high IQA metric value can guarantee high performance in the achievement of computer vision tasks [152].

5.2.2. Underwater image enhancement methods

A number of image enhancement methods for underwater applications has been applied in this work, including both physics-based and neural-network-based ones. Fusion [4] is an effective technique that derives two images from a white-balanced version of the original, degraded input image and then it merges them based on a multi-scale fusion algorithm. Cast-GAN [151] develops an underwater image enhancement method based on a generative adversarial network. Cast-GAN uses the trained generator to remove the color cast from underwater images, without distorting the color of water regions. Finally, WaterNET [154] proposes an enhancement framework based on a convolutional neural network. More specifically, in this technique, three versions of an underwater degraded image are generated by applying white balance [5], histogram equalization [316], and gamma correction [150] algorithms to it. Three confidence maps are extracted accordingly to these three versions of the original image by using a convolutional neural network. In this way, the neural network architecture learns three confidence maps that are subsequently used to combine the three versions into an enhanced result.

5.3. Proposed selection scheme

The selection pipeline is a classification model that relates an image with the best image enhancement technique to be implemented (Fig. 5.1). The approach is based on the Support Vector Machines (SVM) algorithm [38], which is a machine learning technique that maps the data to a higher dimensional space by using a so-called kernel function and searches for a hyperplane to distinctly separate the data points. The data points that are nearest to the hyperplane and are used to make the decision are called support vectors. Some examples of kernel functions for two data points (x_i, x_j) are the linear kernel:

$$K(x_i, x_j) = x_i^{\mathsf{T}} \; x_j + c, \tag{5.1}$$

the polynomial kernel:

$$k(x_i, x_j) = (\gamma x_i^{\mathsf{T}} \; x_j + c)^d, \; \gamma > 0, \tag{5.2}$$

and the radial basis function (RBF)

$$k(x_i, x_j) = e^{-\gamma \| x_i - x_j \|^2}, \; \gamma > 0, \tag{5.3}$$

where γ, c are kernel parameters and d is the degree of the polynomial.

5.4. Methodology

In this section, the implementation of the proposed framework is discussed. The dataset generation process, as well as the procedure followed for the CNN training and the implementation of the selection pipeline are explained.

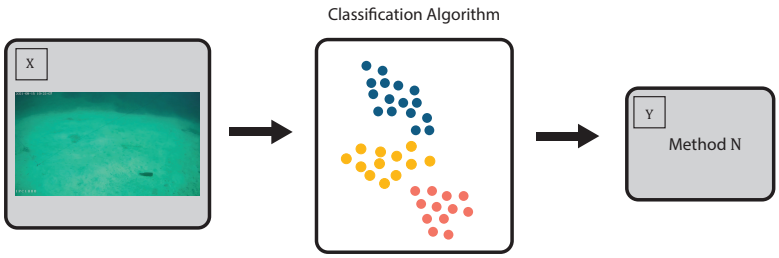


Figure 5.1: Selection pipeline concept.

5.4.1. DATASET SYNTHESIS

In order to generate our dataset, two main aspects are taken into account:

- A broad range of underwater scenes, diverse characteristics of quality degradation, as well as a great variety of image content should be considered.

- The ultimate objective of this study is to assist in the task of underwater litter detection. Hence, the image data contain observations of 4 categories, namely animal, plant, Remotely Operated Vehicle (ROV), and trash.

The dataset is synthesized by real-world underwater images that were collected from different sea areas and by different cameras. More specifically, 300 images come from TrashCan public dataset [109]. TrashCan is sourced from the J-EDI (JAMSTEC E-Library of Deep-sea Images) dataset [236], by the Japan Agency of Marine Earth Science and Technology (JAMSTEC) and it contains videos from ROVs, largely in the sea of Japan. The visibility conditions of TrashCan images are considered as good. Furthermore, 300 additional images are generated from measurements in the sea area by Marseille, France. The visibility conditions in this area are worse compared to TrashCan.

Moreover, 600 images are taken from Dubrovnik, Croatia. The sea areas there have in general very clear visibility conditions. Two experiments were conducted with different hardware and camera equipment, leading to the collection of 300 images from each experiment. The two experiments were different in terms of various conditions such as water conditions and equipment involved. As a result, the first 300 images showcase different characteristics in comparison to the remaining 300 images. We have thus constructed a diverse dataset of in total 1200 real-world underwater images, which is used in order to conduct our comprehensive study with regards to analyzing the image enhancement techniques.

We then implement the image enhancement techniques to the raw underwater images and we generate an image dataset for each implemented technique (Fig. 5.2). In total, we generate four datasets, i.e. raw, Fusion, Cast-Gan, and WaterNET datasets.

5.4.2. Convolutional neural network training

Considering our purpose to analyze underwater image enhancement methods with respect to their impact on the performance of computer vision tasks, we consider a deep-learning-based image classification approach, where the images are fed to a CNN, which in turn predicts their category. Our aim is to train a neural network capable of classifying each sample among the 4 available categories. It is a common practice to initialize the network's parameters using the weights as derived from training on another relatively similar task. In our case, we use the weights that are pretrained on the Microsoft Common Objects in Context (COCO) dataset [162]. COCO is a large-scale image dataset containing 328,000 images of everyday objects and humans. The dataset contains annotations that are widely used to train machine learning models to recognize, label, and describe objects.

Regarding the CNN architecture, You Only Look Once (YOLO)v6 [155] is selected, which is one of the state-of-the-art CNN architectures. YOLOv6 is chosen since it is a real-time architecture with not only large accuracy but also fast inference speed. It is a single-stage object detection framework mostly focusing on industrial applications, with hardware efficient design and better performance than its predecessors [**Sultana2020??**].

In order to evaluate the image enhancement methods, the following two experiments are performed:

Training a CNN for each processed dataset

As a first step, we train YOLOv6 on each one of the four processed image datasets (Fig. 5.2). Having the four CNN networks, we aim to evaluate the impact of each enhancement technique on the feature space representation and its resulting effect on the learning process.

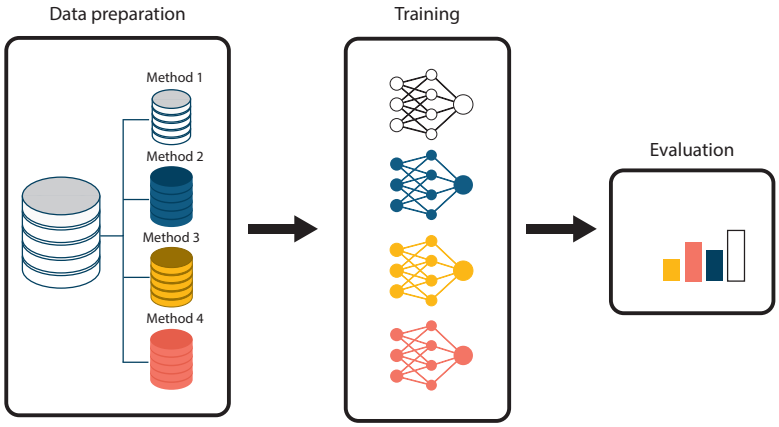


Figure 5.2: Training a CNN architecture for each processed dataset.

TRAINING A CNN ON RAW DATA

In the second experiment, we train YOLOv6 on raw TrashCan data and then we evaluate it on each of the four datasets. With this experiment, we aim to investigate whether a technique can operate as a post-processing step, allowing to employ pre-trained neural networks from other applications. This experiment aligns with enhancing human-based perception on image classification. However, neural-network-based perception vastly depends on the data that it has been trained on, and hence, we want to investigate whether shifting the data distribution might deteriorate the performance.

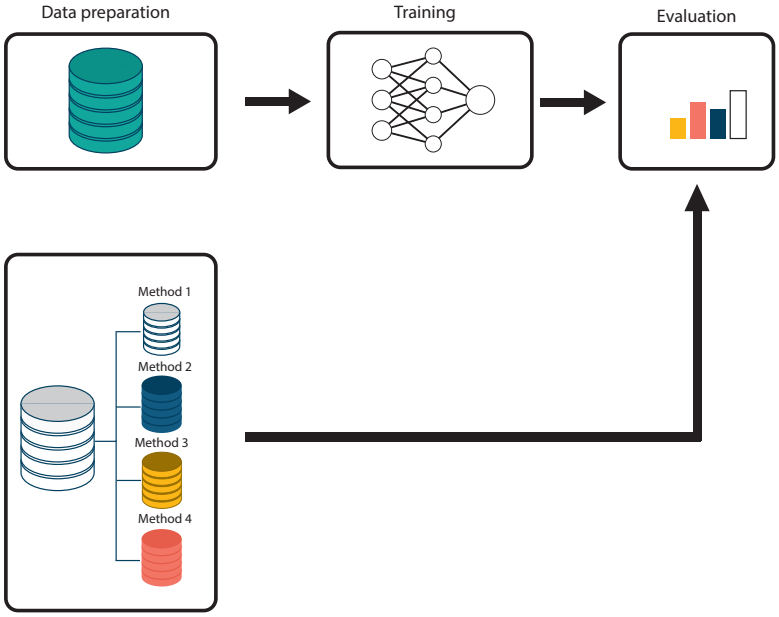


Figure 5.3: Training a CNN architecture on raw image data and evaluating it on enhanced image data

5.4.3. SELECTION FRAMEWORK

Before training the classification model, we implement data augmentation to the image samples, in order to enlarge our dataset. The original dataset contains around 230 images coming from the validation set of the previously described dataset. The augmentation that is performed includes the implementation of image transformations such as rotation, horizontal flip and transpose operations.

Based on the trained CNN architectures and the classification achieved per image, the multi-class SVM is applied to relate images with the best enhancement technique. The implementation is based on the Python library scikit-learn. We explore two variations of this architecture by adjusting the input features provided to the algorithm in order to learn the mapping between an image and the best

enhancement approach: (i) In the first variation, the input features used are the pixel values corresponding to the hue (Hue), saturation (Saturation), and brightness (Value/Brightness) channels. (ii) In the second case, we perform a training of the SVM architecture by adopting the IQA metrics (UCIQE, UIQM, and CCF) as features for the classification task. In this way, we perform a faster training than in the first architecture, since the three IQA values require much less computational effort to be processed than the pixel values of the HSV image channels. The dataset is divided using the ratio of 80:20, where 80 % is for training and 20 % is for testing.

5.5. EXPERIMENTAL RESULTS

Our aim is to construct a mapping between a given image and the respective most suitable image enhancement method, considering the classification performance. For this reason, the predicted classification score and the respective difference with the ground truth per image is extracted. In other words, the predicted labels for each image are compared with the ground truth. The result is also compared with the predicted labels of the image processed by the other methods. For the images whose labels correspond to the correct ground truth values, the confidence score is compared with the other enhancement methods in order to validate which one ensures the highest confidence. An image enhancement technique is regarded as the best performing one per each image if it yields the highest confidence, provided that this confidence is greater than the confidence of the second best performing technique with a threshold of 2%. If this threshold difference is not met, then both of the highest scoring techniques are considered as best performing ones. The results are depicted in Fig. 5.4 and Fig. 5.5. More specifically, Fig. 5.4 depicts the number of images on which each enhancement method reaches the highest classification score while considering the 2% difference threshold. This figure corresponds to the first experiment presented in Section 5.3. We can conclude that the method Fusion outperforms the rest of the methods, while WaterNet is the second best performing method respectively. Finally, CastGan performs the worst among the four methods.

In addition, Fig. 5.4 depicts the results obtained from the second experiment in Section 5.3. From this figure, it can be observed that the implementation of no enhancement (i.e. just raw image data) appears to perform the best, compared to any other image enhancement method. This can be explained by the fact that the training of the CNN is done on solely raw data. As a result, the network has learned to best distinguish the features belonging to each category, when the data are in raw form. WaterNet is the second best performing approach, followed by CastGan. It is also noticeable that Fusion achieves a low performance at this experiment, contrary to the previous one where it outperformed the other methods.

Furthermore, it should be noted that the score using IQA metrics is not necessarily proportional to the classification score. As shown in Fig. 5.7, the CastGAN appears to outperform the other methods regarding the UCIQE metric. Nevertheless, this technique is not one of the top performing ones with regards to the classification performance. At the same time, Fig. 5.6 depicts the performance score per image among the examined methods in a descending order. As can be seen, some methods

might reach quite a high classification score for only a few images, while achieving quite a low performance for the majority of the images.

The classification results are presented in Table 5.1, where the F1-score metric for the two SVM architectures is presented. Note that in this case, the dataset size limits the possibility to achieve higher performance in this four-class classification problem. Hence, in our future work, we will consider larger-size datasets. The code for the developed architecture is available at https://github.com/athinail/IQA_processing.git.

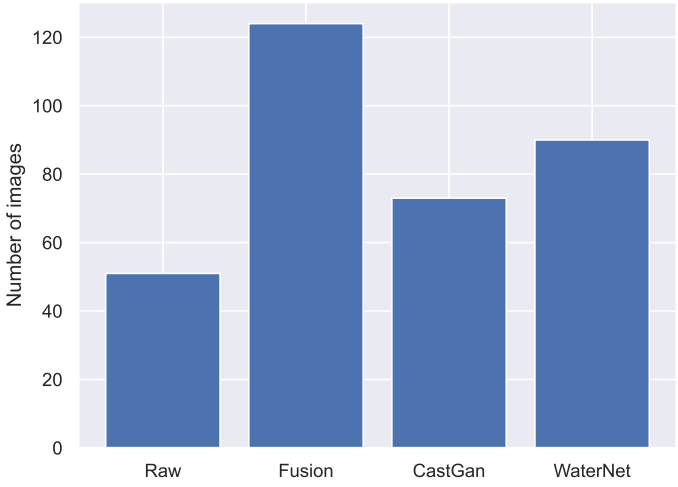


Figure 5.4: Image enhancement methods evaluation based on four CNN architectures trained on each processed image dataset.

Table 5.1: Performance of classification

f1-score	
SVM Classifier - HSV channels as features	41 %
SVM Classifier - IQA metrics as features	46 %

5.6. CONCLUSIONS AND FUTURE WORK

In this chapter a framework is introduced for automating the selection process of the most appropriate underwater image enhancement method to be implemented on an image and the impact of these methods on the performance of classification

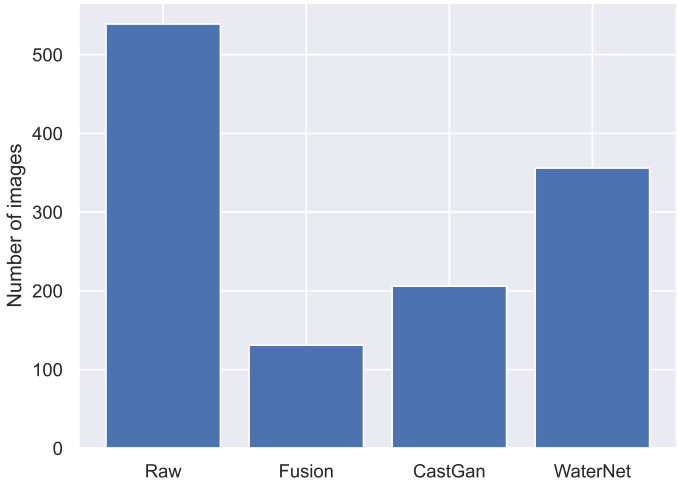


Figure 5.5: Image enhancement methods evaluation based on a CNN architecture trained on raw image data.

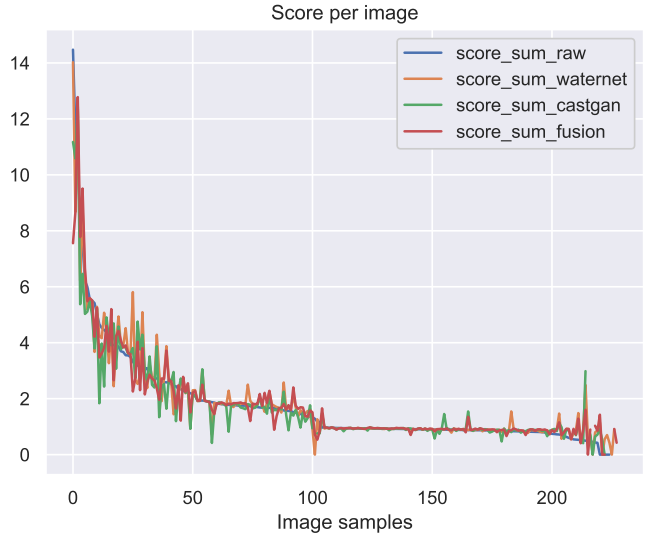


Figure 5.6: Classification score among methods.



Figure 5.7: UCIQE metric per image for each method.

task is explored. The results show that there is a distinction between how the human eye perceives the image enhancement to more easily classify underwater objects and how a trained neural network increases its classification performance by the implementation of various techniques. A selection pipeline is also introduced based on a supervised classification model that performs a mapping between images, their IQA metrics, and their likely optimal enhancement technique.

Future work will focus on the expansion of the dataset to further validate the proposed scheme. An additional important step should concern the implementation and analysis of more image enhancement methods. Furthermore, analyzing the sensitivity of the YOLOv6 neural network employed for classification constitutes an important direction for future research. Alternative architectures, such as other CNN models or transformer-based models, should be evaluated with respect to their performance with image enhancement methods. A comparison of a range of classifier architectures will help assess the generalizability of the enhancement selection method. Finally, an interesting future research direction should concern the implementation of enhancement methods on parts of an image instead of the total image. This could be achieved if, for example, an image is split into a number of areas, and for each area, the most appropriate image enhancement technique is implemented accordingly, instead of applying solely one technique on the total image. In this way, if only parts of an image are used, and in each of these parts a different technique is applied, even higher performance could be achieved, since the texture of an image usually differs along it and hence, a technique can have a varying performance among the different image areas.

6

Enhancing Underwater Robotics Computer Vision for Autonomous Marine Litter Detection

Marine litter pollution is a pressing environmental issue, impacting marine life. The SeaClear 1.0 and SeaClear 2.0 projects funded under the Horizon Europe Programme address this problem using Autonomous Underwater Vehicles (AUVs) equipped with computer vision to detect and remove litter. However, the scarcity of underwater data poses challenges for current models, leading to limited generalization. This study explores strategies to enhance the performance of underwater detection models using state-of-the-art You Only Look Once (YOLO) models (YOLOv6 - YOLOv8). The results show that underwater-specific pretraining and merging multiple underwater datasets improve performance and generalizability, with YOLOv8 achieving the best results. However, even with the merging of underwater datasets, a sufficiently large dataset representing the entire underwater world is not yet achieved. Consequently, the creation of a model with high generalization capability across all environments, attaining high accuracy (>80% mean Average Precision (mAP)@0.5), appears unlikely. To address this, we propose "specialist" models optimized for particular underwater environments while remaining adaptable to diverse locations of the same environmental type. In our case, the models specialize in low versus high turbidity level environments. This approach yields high accuracy (>80% mAP@0.5) while maintaining good generalization within the specified subset of environments.

This chapter is based on the paper: E. P. M. Troquay, A. Ilioudi, K. ter Burg, A. Dabiri, and B. De Schutter. "Enhancing underwater robotics computer vision models and environmental generalization for autonomous marine litter detection". In: *IEEE Journal of Oceanic Engineering* (Apr. 2024). under review.

Author contributions statement: A. Ilioudi contributed to the conceptualization and methodology and

6.1. Introduction

Marine pollution is one of the major current global environmental problems: it poses a growing threat with the potential to damage the natural ecosystem, the economy, and even human health, due to the pervasiveness of degrading plastic as documented by various studies [55, 78, 148, 272].

A significant number of studies predict an increasing influx of waste into water bodies each year [22, 123], and without drastic measures, this influx is projected to surpass the weight of fish by 2050 [287]. In response to this issue, initiatives like Ocean Cleanup[1] have been initiated to address the problem by cleaning the water surfaces. However, multiple field studies [39, 40, 56, 277], and recent models and global simulations [141] have revealed that a substantial proportion of marine litter resides beneath the water surface, sinking through the water column to the seabed.

Since human diving operations are inefficient and unrealistic for long-term deep-sea exploration [224], it becomes evident that an autonomous cleaning system is highly valuable. The described vast amount and widespread distributions of plastic in water bodies emphasize the need for autonomous systems with computer vision, given the nearly impossible task of manual detection and removal of litter (e.g., with divers). Therefore, the goal of the SeaClear 1.0[2] and SeaClear 2.0 initiatives is to detect and remove litter in various marine environments ranging from harbors, over rivers to oceans, using autonomous underwater robots and object detection, while preventing harm to marine life. Underwater object detection is one of the most challenging research topics in the field of computer vision [297]. To ensure the effectiveness of autonomous cleaning systems, it is critical to establish ambitious performance benchmarks. Hence, the SeaClear project has set a target of 80% mAP[3]@0.5 over all classes. The aim of this research is therefore focused on improving the underwater object detection for the SeaClear initiative, and similar (automated) underwater detection systems.

The current status of underwater object detection research highlights two crucial points of improvement. First, there is a need to increase the amount of labeled data to capture a broader range of underwater conditions, aiming for a representative sample of the diverse environments encountered. Second, incorporating more recent state-of-the-art architectures is essential to enhance the performance of underwater

reviewed the manuscript.

[1] The Ocean Cleanup is a nonprofit organization that aims to develop advanced technologies to rid the world's oceans of plastic pollution. Founded in 2013 by Dutch inventor Boyan Slat, the organization has developed a unique system that uses natural ocean currents to passively collect plastic debris in floating garbage patches. The system consists of a long floating barrier that collects plastic debris and a central platform that acts as a processing plant to extract, sort, and recycle the collected plastic.

[2] SEACLEAR — short for SEarch, identificAtion and Collection of marine Litter with Autonomous Robots — is a project funded under the Horizon Europe Programme that aims to solve marine litter pollution using underwater robotics and artificial intelligence.

[3] mAP@0.5, also called mAP^{50}, stands for mean Average Precision at Intersection over Union (IoU) threshold of 0.5. An IoU threshold of 0.5 requires a relatively significant overlap between the predicted bounding box and the ground-truth bounding box, at least 50%, for the detection to be considered correct. Higher IoU thresholds (e.g., 0.7 or 0.75) are also used for stricter evaluation criteria. However, mAP@0.5 serves as a widely used standard metric.

object detection systems.

A critical factor for successful deep learning-based detection of underwater objects is the presence of diverse and realistic underwater environments, encompassing a wide range of target objects and object poses within the dataset. However, training deep neural networks necessitates a considerable number of samples, while obtaining an extensive collection of underwater image data can be challenging. Due to the complex underwater environment and the high requirements for underwater imaging equipment, it is more difficult to obtain high quantity (and high-quality) underwater images than to obtain atmospheric optical images. As a result, the underwater datasets are small in size and/or suffer from imbalanced classes of different underwater targets, leading to potential overfitting issues. This limitation severely hinders data-driven deep learning models from achieving satisfactory results in underwater object detection [210, 247, 248]. Moreover, numerous studies have indicated that the current datasets lack a sufficient quantity of underwater images to achieve high accuracy (>80% mAP@0.5) in the underwater environment [21, 36, 50]. Addressing these challenges and enhancing the availability and diversity of underwater image datasets remain vital for advancing the performance of underwater object detection models [12, 160, 203].

When the acquisition of larger datasets is not feasible, a common strategy to address the limited size of training data in underwater object detection is transfer learning [288]. This approach involves leveraging knowledge gained from one task to enhance the performance on a new problem of interest [199]. By reusing a pretrained network that has learned from a large dataset like Microsoft Common Objects in Context (COCO) [162] or ImageNet [45], prediction accuracy can be improved, even when the dataset for the new task is relatively small. For instance, transfer learning has shown notable enhancements in classification performance for above-water surface floating marine litter test images [183]. However, in many underwater object detection studies, pretrained weights from the COCO dataset are utilized, which lacks underwater images of marine life or litter. As a result, it becomes interesting to investigate the impact of using a large pretraining dataset that includes relevant categories, such as ImageNet, for underwater marine litter detection. Such investigations would provide insight into the benefits of using more suitable pretraining datasets in improving the performance of underwater object detection models.

Furthermore, the limited amount of data poses a challenge in developing models that can generalize well to different underwater environments [126, 297]. The environmental generalization capability refers to a model's ability to perform effectively in diverse environmental conditions, such as varying water turbidity [126]. As an example, [160] demonstrated that a trained model, that performed reasonably well in one location, exhibited a significant deterioration in performance when applied to an unseen location, resulting in a decrease in accuracy from 68.7% to 54% [160]. Although generic and marine litter-specific underwater datasets exist [11, 66, 110, 124, 208, 213, 262], they are not as extensive as COCO or ImageNet datasets. Consequently, building a model that can realistically represent the complexities of the underwater world and generalize to different environments becomes challenging.

However, research has shown that the generalization performance can be enhanced by including images from datasets collected at various locations in the training dataset [160].

Given the current size and diversity of existing underwater marine life and litter datasets, even the combination of them may not match the scale of ImageNet and thus may not fully represent the entire underwater world. As a consequence, a model designed for generalization across all underwater environments and conditions is likely to exhibit lower performance compared to models trained to specialize more towards a certain type of underwater environment, which likely requires less data. These "specialist" models excel in specific underwater environments, leveraging their expertise to adapt to certain underwater conditions such as differences in water turbidity levels. The degree of water turbidity can significantly alter the appearance of underwater environments, necessitating object detection models to adapt to extract effective object features. Despite their specialization, these models maintain the ability to generalize across different locations, as long as they share the same type of underwater environment. In situations where high accuracy is of paramount importance, creating multiple specialized models tailored to specific underwater environments might be a preferred strategy. Therefore, exploring and evaluating the performance of both generalizing models and specialized models holds great value in underwater object detection research.

6.2. DATA PREPARATION AND MODEL TRAINING

6.2.1. DATASETS

For the experiments, we curated a diverse set of marine litter and marine life datasets, each chosen for its relevance and suitability in evaluating the performance of the models under study. The selected datasets include SeaClear [52], TrashCan (Instance 1.0) [109], Imagenet-1K [45], RUOD [64] and Brackish [207], as summarized in Table 6.1.

Several other datasets, as listed below, were excluded from our study due to various reasons:

Table 6.1: Summary of bounding box annotated image datasets for underwater marine life and/or litter detection.

Dataset	Environment (Class examples)	Image number	Class number	Year
SeaClear	Mixed clear and brackish under water (Bottle_plastic, plant, animal_urchin, rov_tortuga)	8610	40	2022
TrashCan-Instance 1.0	Medium turbidity underwater (Bag, clothing, rope, wreckage, etc.)	7212	22	2020
ImageNet-1K	Various underwater (jellyfish, brainco-ral, plasticbag)	1331167	1000	2012
RUOD	Clear underwater (Fish, sea urchins, corals, starfish, sea cucumbers, etc.)	14000	10	2022
Brackish	Brackish underwater (Fish, crabs, starfish, jellyfish, shrimp)	14518	6	2019

- Datasets without bounding box annotations or any annotations at all (e.g., WildFish [315], Deep-Sea Debris database [124]).
- Datasets consisting solely of above-water trash, which do not represent the underwater environment (e.g. TACO [214], PlastOPol [36]).
- Datasets that represent smaller subsets of the TrashCan [109] dataset (e.g., Trash-ICRA19 [65], DeepPlastic [262], Deep-Sea Debris database [124]).
- Datasets with a relatively limited diversity of marine life (e.g., URPC datasets [91], UDD [166], DUO [165], UODD [129]).

In what follows, we discuss the selected data sets in more detail.

SEACLEAR

The SeaClear dataset [52], developed by the SeaClear consortium, encompasses 8610 bounding-box annotated images captured using Remotely Operated Vehicles (ROVs) at multiple underwater locations, including the coast of Lokrum and Bistrina in Croatia and in Marseille, France. This dataset features diverse classes, portraying various marine life, litter, and the ROV itself.

The images in the SeaClear dataset offer a comprehensive representation of both clear and brackish underwater environments, with clear environments primarily from the Lokrum location and brackish environments from Bistrina and Marseille. The inclusion of images from different underwater conditions, characterized by varying turbidity levels, provides a realistic and challenging test bed for evaluating various underwater object detection architectures.

In this study, the SeaClear dataset is leveraged to validate the effectiveness of our models (YOLO neural networks), as the data aligns closely with the real-world conditions typically encountered in diverse underwater operations. By evaluating the performance of the models on the SeaClear dataset, we gain valuable insights into their capability to detect and classify marine objects and litter in various underwater environments.

TRASHCAN

The TrashCan [109] dataset consists of 7212 bounding box annotated images, encompassing diverse observations of trash, ROVs, and a wide variety of undersea flora and fauna. Two versions of the dataset exist: TrashCan-Material and TrashCan-Instance. For our study, we exclusively utilize the TrashCan-Instance version which focuses on object-level annotations, as our research aims to identify and categorize different underwater object types rather than distinguishing between material types.

The images in the TrashCan dataset originate from the J-EDI (JAMSTEC E-Library of Deep-sea Images) dataset [124], collected by the Japan Agency of Marine Earth Science and Technology (JAMSTEC). These images are predominantly sourced from videos captured by ROVs operated by JAMSTEC since 1982, primarily in the sea of Japan.

Our selection of the TrashCan dataset is driven by multiple factors. Firstly, it is the largest underwater marine litter dataset to date. Additionally, the dataset's curation and proven usage in other scientific studies reinforces its credibility and facilitates a comparison with the newly created SeaClear dataset. This enables a thorough evaluation of both the quality and the challenges of both datasets in underwater litter detection.

ImageNet-1K

ImageNet-1K [45], the bounding box annotated version of the ImageNet dataset, comprises around 1.2 million images, categorized into 1000 classes. These images originate from diverse online sources and are labeled with bounding boxes, spanning a wide spectrum from everyday objects to complex scenes and concepts.

In contrast to the COCO dataset, ImageNet-1K's class repertoire includes various marine life, plants, and common marine litter objects such as plastic bottles and bags. For this research, we selected 58 classes that specifically depict these target objects, intending to perform additional pre-training of models already pre-trained on COCO.

The training set for each class contains a minimum of 421 and a maximum of 704 annotated images, with an average of 529.43 (std ± 70.09). Consequently, the final size of the training and validation sets stands at, 30707 and 2900 images, respectively. This subset of the ImageNet-1K dataset serves as a foundation for investigating the impact of extra-pretraining on relevant target objects, such as underwater marine life and litter.

RUOD

The RUOD [64] dataset comprises 14000 bounding-box annotated, high-resolution images, showcasing a rich variance of marine objects and clear underwater environments. It was created to address the limitations of existing underwater object detection datasets, which often suffer from a scarcity of images, limited categories, low resolution, and insufficient representation of environmental challenges.

RUOD is an underwater, natural-light dataset carefully curated for underwater detection. The dataset is a collection of images from various websites, supplemented by a few images from URPC2020 [91]. The dataset focuses on the 10 marine objects with the highest frequency of occurrence in day-to-day life, presenting a diverse range of types, appearances, and scales.

Due to its distinctive features, particularly the abundant variety of marine objects and clear underwater environments, RUOD has been selected to enhance the robustness of our model not only in general underwater scenes but also specifically in clear underwater environments.

Brackish

The Brackish [207] dataset encompasses 14518 images, out of which 12444 contain objects of interest with bounding box annotations. These images feature a diverse

collection of marine objects captured in brackish or unclear water conditions, often characterized by varying but mostly high turbidity.

The dataset is captured using stationary cameras positioned at the bottom of Limfjorden in Denmark, approximately nine meters below the water surface. This setup ensures the portrayal of authentic underwater conditions, reflecting the real-world challenges of underwater litter detection in brackish environments.

Similar to the RUOD dataset, the Brackish dataset includes the most common marine animals found in day-to-day life, represented across six distinct classes.

We have deliberately selected the Brackish dataset for this study to not only bolster the overall robustness of our model through its abundant training data but also to specifically address the complexities of underwater object detection in brackish/unclear water with varying turbidity.

To ensure consistency and comparability across all datasets, a standardized data preprocessing pipeline was applied. The original class labels in each dataset were mapped to a unified set of four classes: ROV, plant, animal, and trash.

The training and validation split for most datasets followed an 80-20 ratio. An exception was made for the ImageNet-1K dataset, where we preserved the conventional split of approximately 1.2 million images for training and 50000 for validation, as established by the ImageNet Large Scale Visual Recognition Challenge [229]. This setting selection was based on the objective of maintaining consistency with historical benchmarks and facilitating comparison with prior studies, which widely employ this established split. Furthermore, for the Brackish dataset, which was split 80-10-10 (train-validation-test), the validation and test datasets were merged to create the validation set. Lastly, a key consideration to prevent time-dependent data leakage, especially in datasets containing videos, involves the careful partitioning of entire videos into either the training or validation set. By treating complete videos as cohesive entities during the split, we mitigate the risk of overlapping frames between the two sets and safeguard against inadvertent contamination of the validation set with information from the training set. By preserving the independence and integrity of each set throughout the training and validation processes, we enhance the robustness and reliability of our model evaluation.

6.2.2. OBJECT DETECTION MODELS

In object detection, YOLO models are extensively used due to their balance of speed and accuracy, which is essential for real-time applications. This study aims to leverage state-of-the-art YOLO models: YOLOv6 [155], YOLOv7 [281], and YOLOv8 [130]. These versions have been chosen for their enhanced capabilities to efficiently and accurately handle challenging conditions that are typically encountered in underwater environments.

Table 6.2 presents the relevant benchmark attributes of these selected models, providing insight into their respective performance characteristics. Notably, YOLO models are renowned for surpassing other object detectors in terms of speed, measured in frames per second (FPS), and accuracy in terms of mAP, as evaluated on the COCO benchmark dataset. Additionally, YOLO models offer compact model sizes

Table 6.2: Specifications of YOLO models.

Model	Input Size (pixels)	$mAP^{0.5}_{val}$(%)	$mAPval^{0.5-0.95}$(%)	Params (M)	FPS (bs=1)	FLOPs (B)
YOLOv6n [155]	640	51.2	37.5	4.7	779	11.4
YOLOv6s [155]	640	60.4	45.0	18.5	339	45.3
YOLOv7-tiny [281]	640	55.2	37.4	6.2	424	13.7
YOLOv8n [130]	640	52.5	37.3	3.2	1163	8.7
YOLOv8s [130]	640	61.8	44.9	11.2	925	28.6

in terms of the number of parameters (M). These features make them well-suited for real-time applications.

For this study, particular emphasis was placed on the smaller available models, namely the nano (n), tiny (t), and small (s) versions of each YOLO version. The trend in newer YOLO iterations suggests that these smaller models not only have fewer parameters but also exhibit faster training and real-time performance while demonstrating improved accuracy.

6.2.3. Model Training

The training of models with the selected datasets was conducted on the computational resources of the DelftBlue supercomputer - GPU partition, provided by the Delft High-Performance Computing Centre. DelftBlue is equipped with NVIDIA V100S GPUs, where one GPU was requested per job, allowing high-performance computing for our training tasks.

Validation and inference processes were performed locally on a MacBook Air (Retina, 13-inch, 2018), which features an Intel UHD Graphics 617 GPU with 1.5GB of shared memory. The MacBook Air also has an 8th-generation 1.6GHz dual-core Intel Core i5 processor and 8GB of 2133MHz LPDDR3 memory (RAM).

The YOLO models were trained for 100 epochs, using a batch size of 32 and an image size of 640 pixels. Default hyperparameter configurations were applied, ensuring consistency for comparison across the trained models. These training parameters were chosen to strike a balance between computational efficiency and model convergence, enabling the YOLO models to effectively learn and generalize to the underwater litter detection task.

The evaluation consists of four parts, each focusing on specific aspects of underwater object detection using YOLO architectures:

- **PART I**: In this part, a YOLOv6n model pretrained on COCO was trained separately on the TrashCan and SeaClear datasets. Additionally, a cross-dataset evaluation was conducted, where the model was trained on the entire dataset of TrashCan or SeaClear and validated on the entire other dataset. Finally, the architecture was trained on the merged TrashCan and SeaClear dataset.

- **PART II**: This experiment involves additional pretraining of a YOLOv6n model on selected underwater classes from ImageNet-1K. Subsequently, the model

was trained on either the TrashCan, SeaClear, or the merged TrashCan SeaClear dataset.

- **PART III**: For this part, the RUOD, Brackish, TrashCan, and SeaClear datasets were merged. Five YOLO architectures (YOLOv6n, YOLOv6s, YOLOv7-tiny, YOLOv8n, and YOLOv8s), pretrained on COCO and ImageNet, were trained on the merged dataset. Besides the validation on the merged validation set, the models were also evaluated solely on the SeaClear dataset.

- **PART IV**: The SeaClear dataset was split into clear water and brackish water portions, including respectively 1552 and 7058 images, based on specific locations known for their water clarity. Images with artifacts, such as dirt on the camera lens, were removed[4]. The clear portion of the SeaClear dataset was then merged with the RUOD and TrashCan datasets, and the brackish portion with the Brackish and TrashCan datasets. The models were then trained on either the merged clear water or merged brackish water dataset. The validation was done with the respective merged clear and brackish validation set. Lastly, the merged clear water dataset was also evaluated on the clear water portion of the SeaClear dataset, and the merged brackish dataset on the brackish portion of the SeaClear dataset.

6.3. RESULTS & DISCUSSION

6.3.1. BASELINE & GENERALIZABILITY

The first two groups of bars in Figure 6.1 illustrate the baseline mAP at 50% IoU[5] (mAP@0.5) results for the TrashCan and SeaClear validation datasets, divided into four classes (ROV, plant, animal, and trash) and the overall average denoted as “all”. The baseline represents a YOLOv6n architecture trained on the COCO dataset with basic training setup. The TrashCan baseline achieved a mAP@0.5 of 68.7%, while the SeaClear baseline performed better at 77.4%.

The third and fourth groups of bars show the cross-dataset evaluation results. The left group presents the performance of the YOLOv6n model trained on SeaClear and evaluated on TrashCan, while the right group demonstrates the reverse scenario. The mAP@0.5 for these cross-dataset evaluations were 6.35% and 17.5%, respectively.

In the fifth group of bars, the YOLOv6n model was trained and validated on a merged dataset containing both TrashCan and SeaClear samples. This merged dataset resulted in a mAP@0.5 of 60.2%.

[4]Since the dirty lens artifacts could influence the model performance, and are unrelated to the environment, we chose to remove these images in this experiment part. Possible effects on the results caused by including or excluding the artifact images are discussed in our GitHub repository.

[5]IoU is an evaluation metric used in computer vision tasks, particularly in object detection and image segmentation. It measures the overlap between the predicted bounding box (or segmentation mask) and the ground truth bounding box (or segmentation mask). The IoU is calculated as the ratio of the intersection area to the union area of the two bounding boxes (or masks), and it ranges from 0 to 1, where a higher IoU indicates better overlap between the predicted and ground truth regions.

The overall observations suggest that the models perform well when trained and validated on the same dataset[6]. However, their performance drastically reduces when tested on a different dataset, indicating limited generalization capability. Notably, the model trained on TrashCan demonstrated slightly better generalization, primarily due to the high performance in the ROV class. The ROV class's consistent appearance across varying water conditions likely contributes significantly to this outcome, as its visual characteristics exhibit minimal variability despite fluctuations in water turbidity. This consistency enhanced the model's ability to generalize effectively for this class. Merging datasets from different environments appears to improve generalization, with the combined mAP@0.5 recovering to around 60%. Nonetheless, the performance of the merged dataset remains lower than training on individual datasets. Additionally, the plant and/or animal classes consistently exhibit the lowest performance across all experiments.

It is worth highlighting that merging datasets with varying environmental conditions indeed aids in enhancing the generalization capability, aligning with prior research [160]. Further exploration in subsequent experiments III-IV will delve deeper into this aspect.

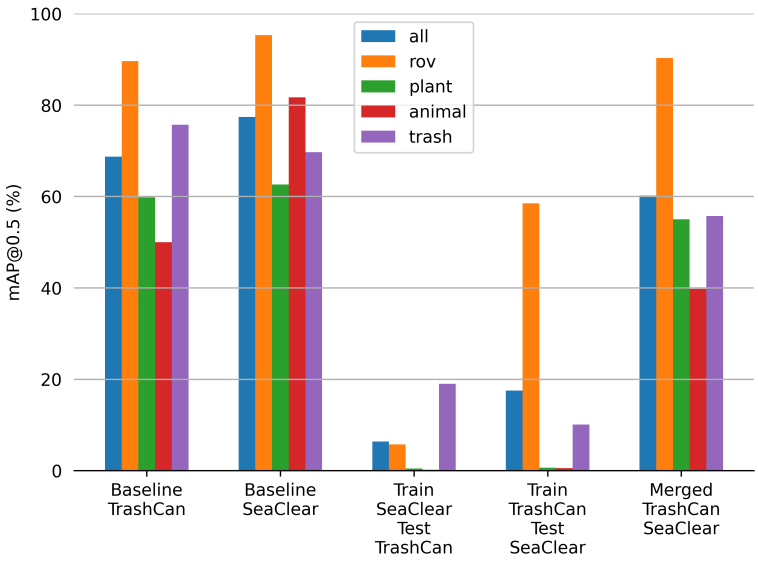


Figure 6.1: Baseline, cross-dataset & merged TrashCan and SeaClear dataset evaluation.

[6]The results of the baseline TrashCan are comparable to that of the results observed at OpenCV [221], where an mAP@0.5 of 72.4% was obtained over all classes

6.3.2. EFFECT OF UNDERWATER SPECIFIC PRETRAINING

In Figure 6.2, the first, third, and fifth groups of bars represent the results of the TrashCan baseline, SeaClear baseline, and the merged TrashCan SeaClear model, respectively. They are compared with the second, fourth, and sixth groups of bars, which show the results of the corresponding models with additional pretraining on ImageNet. The ImageNet pretrained models consistently outperform their counterparts trained solely on COCO. Specifically, the TrashCan baseline achieved an mAP@0.5 of 68.7%, whereas the ImageNet pretrained TrashCan model achieved 70.0%. Likewise, the SeaClear baseline achieved an mAP@0.5 of 77.4%, while the ImageNet pretrained SeaClear model performed significantly better at 84.1%. Furthermore, the Merged TrashCan SeaClear model achieved an mAP@0.5 of 60.2%, whereas the ImageNet pretrained merged TrashCan SeaClear model achieved 62.8%.

The additional pretraining on ImageNet had a minor impact on the TrashCan model, resulting in a slight improvement of approximately 1%. However, for the SeaClear model, there was a notable increase in performance of around 7%, primarily driven by improvements in the plant class. Similarly, the Merged TrashCan SeaClear model showed a moderate improvement of about 3%, mainly attributed to enhanced performance in the animal class. The improved performance in the plant and animal classes can be attributed to the fact that the selected ImageNet classes largely consist of underwater marine life.

Overall, the results indicate that the supplementary pretraining on ImageNet positively influences underwater object detection performance, particularly by bolstering the performance of weaker classes. This is in line with former transfer learning research [183]. As a result of these findings, all models in the subsequent experiments have incorporated additional pretraining on ImageNet.

6.3.3. EVALUATING ENVIRONMENT GENERALIST MODEL PERFORMANCE

The bar plot in Figure 6.3 presents a comparative analysis of three YOLOv6n models: the merged TrashCan SeaClear model (with and without additional ImageNet pretraining) of section I-II, and a model trained and validated on an expanded merged dataset. The first and second groups of bars recap the results of the merged TrashCan SeaClear model, both with and without additional ImageNet pretraining, with an mAP@0.5 of 60.2% and 62.8% respectively. The third group of bars showcases the performance of a YOLOv6n model trained and validated on an expanded merged dataset. This comprehensive dataset encompasses not only TrashCan and SeaClear but also the RUOD and Brackish datasets, which appear to boost the plant and animal classes. The YOLOv6n model trained on this merged dataset achieves a mAP@0.5 of 72.2%, surpassing the performance of models trained solely on TrashCan and SeaClear datasets. From left to right, the mAP@0.5 values are 60.2%, 62.8%.

Comparing the results, it is evident that merging more datasets for training has a more significant impact on the performance of a model compared to additional ImageNet pretraining. The former approach leads to a substantial improvement of approximately 10% in mAP@0.5, while the latter results in approximately 3% improvement.

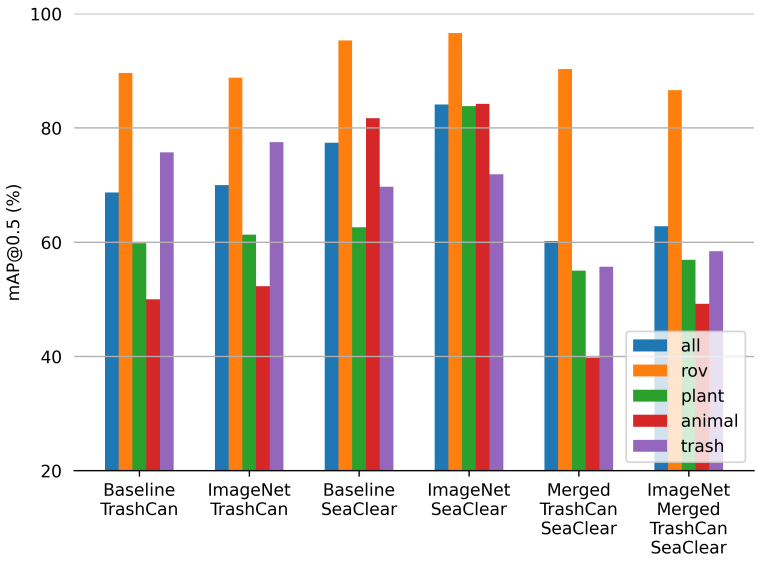


Figure 6.2: Effect of underwater specific pretraining with ImageNet.

Building on these promising findings, we investigate the performance of this larger merged dataset when applied to different YOLO architectures, exploring potential variations in object detection performance.

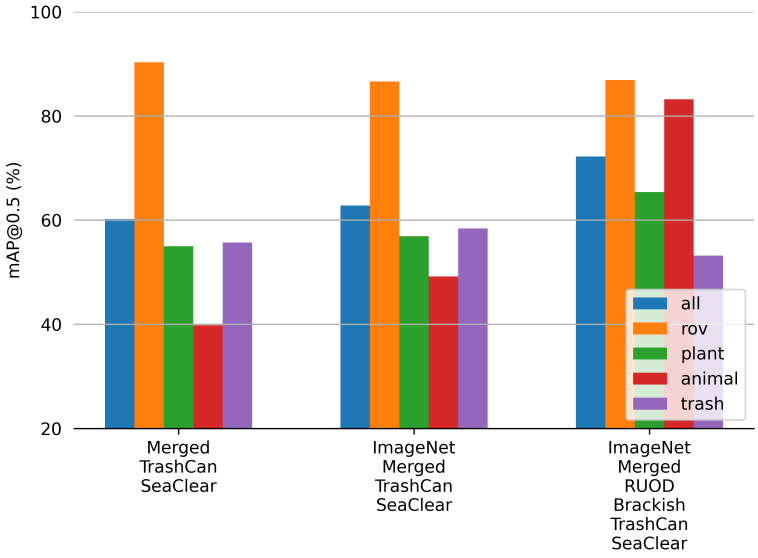


Figure 6.3: Effect of merged datasets on generalization performance.

In Figure 6.4, the effect of training and validating five different YOLO models (YOLOv6n, YOLOv6s, YOLOv7-tiny, YOLOv8n, and YOLOv8s) on the merged dataset containing RUOD, Brackish, TrashCan, and SeaClear datasets is depicted. The mAP@0.5 values over all classes are 72.2%, 73.7%, 72.7%, 73.1%, and 75.5% for the respective models.

Figure 6.5 shows the results of these models when they are specifically validated on the SeaClear dataset. The mAP@0.5 values over all classes are 74.9%, 85.4%, 88.2%, 88.2%, and 94.2% respectively.

The figures illustrate the performance of the models in a mixed environment of clear and brackish water. Overall, an increase in performance is observed when using larger and more recent YOLO versions. Specifically, when validated on the SeaClear dataset, there is a substantial increase in mAP@0.5 around 20% when switching from YOLOv6n to YOLOv8s. However, we observe that the models YOLOv6s, YOLOv7-tiny, and YOLOv8n show less variation and perform roughly the same. As for the merged dataset, the variation between the models is at most approximately 3%, with YOLOv8s being the best performing model.

Additionally, when the models are validated on the data from the SeaClear environment, they generally exhibit higher performance, with most models reaching the target of 80% mAP@0.5. This improvement can be attributed to the addition of the RUOD and Brackish datasets, which positively impacts the performance in the animal and plant classes.

However, when validated on all datasets, none of the models achieve the target of 80% mAP@0.5. This indicates that although the models perform well in the SeaClear environment, they do not generalize as effectively across all other environments.

These findings highlight the importance of considering the specific environment and dataset when evaluating the performance of object detection models, as performance may vary significantly based on the characteristics of the dataset. The latter will be further explored and discussed in the next experiment.

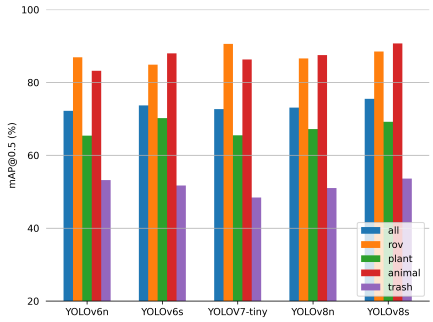


Figure 6.4: Performance of environment generalist models trained with several YOLO architectures, validated on all datasets.

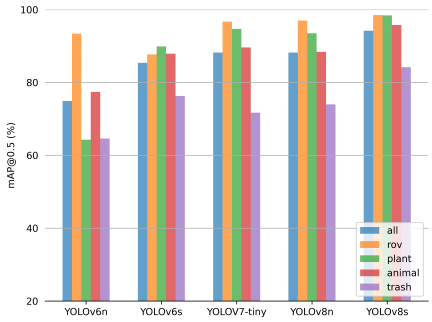


Figure 6.5: Performance of environment generalist models trained with several YOLO architectures, validated on the SeaClear dataset.

6.3.4. Evaluating environment specialist model performance

In Figure 6.6, we observe the effect of training and validating five different YOLO models (YOLOv6n, YOLOv6s, YOLOv7-tiny, YOLOv8n, and YOLOv8s) on a merged dataset containing the RUOD and TrashCan datasets, along with a portion of the SeaClear dataset containing clear water images. The mAP@0.5 values over all classes are 75.6%, 80.5%, 78.8%, 80.1%, and 81.8% for the respective models.

Figure 6.7 shows the results of these models when they are specifically validated on the clear water portion of the SeaClear dataset. The mAP@0.5 values over all classes are 66.9%, 79.3%, 80.1%, 79.1%, and 83.6% for the models, respectively.

The figures demonstrate the performance of the models in clear water environments. Similar to the observations in Figure 6.4 and Figure 6.5, using larger and more recent YOLO versions results in an increase in performance. Moreover, when validated on the clear water portion of the SeaClear dataset, there is again a more substantial increase in mAP@0.5 (compared to the merged dataset in Figure 6.6), which corresponds to approaximately 16% when switching from YOLOv6n to YOLOv8s. Conversely, the models YOLOv6s, YOLOv7-tiny, and YOLOv8n show less variation and perform comparably. As for the merged dataset, the variation between the models is at most approximately 4%, with YOLOv6n and YOLOv7-tiny being the least performing models, while YOLOv8s performs the best.

Furthermore, when the models are validated on the data from the clear water SeaClear environment, they generally exhibit a slightly lower performance compared to the overall clear water dataset performance. It appears that the addition of the RUOD dataset does not boost the performance in the animal and plant classes for the clear water SeaClear environment as strongly as it does for the mixed SeaClear environment. Nonetheless, in both scenarios, some models reach the target of 80% mAP@0.5, with YOLOv8s achieving this threshold in both evaluations.

These results indicate that the models not only perform well in the clear water environment of SeaClear but also demonstrate the ability to generalize to other clear water environments from the other datasets.

In Figure 6.8, we observe the effect of training and validating five different YOLO models (YOLOv6n, YOLOv6s, YOLOv7-tiny, YOLOv8n, and YOLOv8s) on a merged dataset containing the Brackish and TrashCan datasets, along with a portion of the SeaClear dataset containing brackish water. The mAP@0.5 values over all classes are 81.7%, 87.5%, 84.0%, 86.0%, and 86.8% for the respective models.

Figure 6.9 shows the results of these models when they are specifically validated on the brackish water portion of the SeaClear dataset. The mAP@0.5 values over all classes are 85.6%, 91.1%, 90.8%, 91.5%, and 94.3% for the models, respectively.

Similar to the observations in Figures 6.4 to 6.7, using larger YOLO network architectures and more recent versions increases the performance. Notably, however, the YOLOv6s model shows the highest performance for the merged brackish water datasets. Furthermore, when validated on the brackish water portion of the SeaClear dataset, there is again a more substantial increase in mAP@0.5 (compared to the merged dataset in Figure 6.8), ranging from around 10% when switching from YOLOv6n to YOLOv8s. Conversely, the models YOLOv6s, YOLOv7-tiny, and YOLOv8n show less variation and perform comparably. As for the merged dataset, the variation

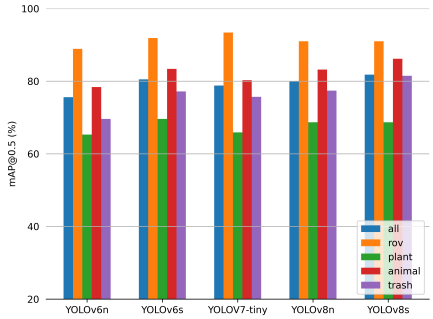


Figure 6.6: Performance of clear water environment specific models trained with several YOLO architectures, validated on all clear water datasets.

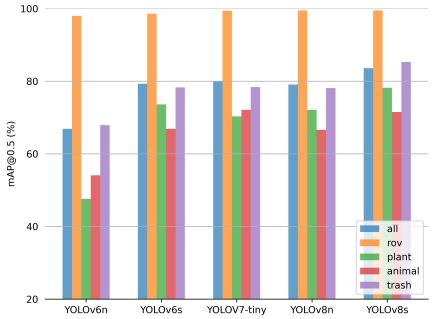


Figure 6.7: Performance of clear water environment specific models trained with several YOLO architectures, validated on the clear water portion of the SeaClear dataset.

between the models is approximately 5%, with YOLOv6n and YOLOv7-tiny being the lowest performing models, while YOLOv6s performs the best.

Additionally, when the models are validated on the data from the brackish water SeaClear environment, they generally exhibit a higher performance, with all models reaching the target of 80% mAP@0.5. This improvement can be partly attributed to the addition of the RUOD and Brackish datasets, which positively impacts the performance in the animal and plant classes. The other part can be attributed to the generally higher-performing trash class. Similarly to the plant and animal classes, the trash class also shows improved performance in merged brackish water environments, where also all models reach the target of 80% mAP@0.5.

These results indicate that the models not only perform well in the brackish water environment of SeaClear but also demonstrate the ability to generalize to other brackish water environments from the other datasets.

Comparing Figures 6.6 and 6.7 with Figures 6.8 and 6.9, it can be observed that models specializing in clear water environments show a lower performance than models specializing in brackish water environments. Several explanations can be considered:

- The portion of brackish water data in the SeaClear dataset is larger than that of clear water data. Out of 8610 images, 7058 come from brackish water environments, while only 1552 come from clear water environments. This larger quantity of brackish water data allows for a more extensive training on this type of environment, potentially leading to improved performance.
- The variety of marine life in the RUOD dataset is larger than that of the Brackish dataset. The RUOD dataset originally contains 10 classes, while the Brackish dataset has only 6. While a larger variety of classes can aid in

generalization, it may also add complexity to the model, potentially impacting performance if certain classes are not adequately represented across different environments.

The above observations may also explain the high performance of the models in the mixed SeaClear environment shown in Figure 6.5, as the mixed SeaClear environment dataset predominantly consists of the easier brackish water environment data. Thus, the SeaClear dataset might not adequately represent the variability of a mixed clear and brackish water environment.

Comparing Figures 6.6 to 6.9 with Figures 6.4 and 6.5 we see YOLOv8n emerging as the preferred architecture as it shows the highest performance in nearly every figure, which is in line with earlier research [16, 126, 168, 209, 276].

Comparing these figures, it is also evident that models specializing in either the clean or brackish water environments show higher performance than the generalist models. Depending on the application of the models, one or the other type of model can be preferred. If a widely applicable model without the need for high accuracy is desired, a generalist model suffices. However, if high accuracy is a priority, the use of environment specialist models may be necessary, e.g. to robustly reach the target of 80% mAP for the SeaClear application.

For more detailed information on the evaluation metrics, samples of the images of each dataset, and the mapping of the original classes to the four classes used in this study, we refer to our GitHub repository. Additionally, in the same repository, a link to an inference video example can be found.

6.4. Conclusions & Future Work

In this study, our primary aim was to enhance the underwater detection performance of the SeaClear system and similar robotic underwater detection systems. The SeaClear research group has set the ambitious goal of underwater object detection with a mAP@0.5 of 80% on average over all classes, in order to detect and collect marine litter while ensuring the preservation of marine life. To achieve this goal, we have conducted several experiments to explore strategies not only for increasing underwater object detection accuracy but also for improving the generalization capability of the models across different environments.

Based on the experimental findings, we can conclude that incorporating underwater-specific pretraining and merging datasets from diverse environments significantly improves model performance and generalization capacity. Furthermore, more recent YOLO architecture versions consistently outperform their older counterparts, with YOLOv8 being the most favorable option.

Moreover, our results indicate that models specialized for specific environmental conditions, such as water with high vs. low turbidity, exhibit higher accuracy performance while still retaining their ability to generalize to various environments based on their specialization feature (i.e., turbidity level). However, the choice between using a generalist or specialist model depends on the specific application of the underwater detection system. In the case of a system like that of SeaClear, a

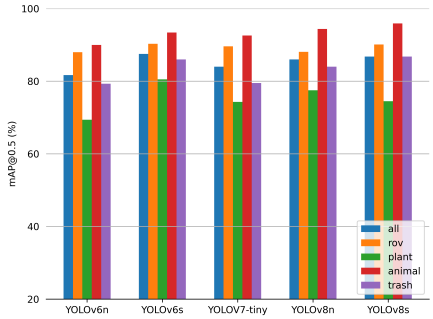


Figure 6.8: Performance of brackish water environment specific models trained with several YOLO architectures, validated on all brackish water datasets.

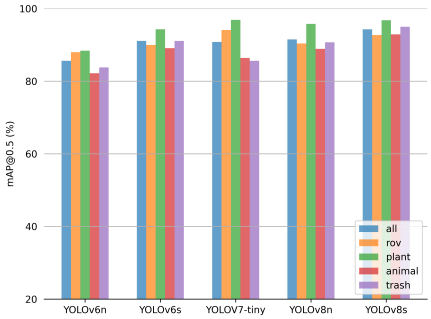


Figure 6.9: Performance of brackish water environment specific models trained with several YOLO architectures, validated on the brackish water portion of the SeaClear dataset.

specialist model with high accuracy is preferred in order to prevent harm to marine life and to effectively remove marine litter.

To develop a high-performing generalist model that effectively represents all underwater environments, it becomes apparent that an ImageNet-sized dataset is necessary for the underwater domain. Future work is encouraged to develop such a dataset, which would greatly facilitate initiatives like SeaClear in developing efficient systems to combat marine pollution.

This research contributes to enhancing the capabilities of underwater object detection. By harnessing the potential of state-of-the-art YOLO architectures and expanding the availability of diverse underwater datasets, this work paves the way for more effective and efficient underwater computer vision systems in the future.

7

Fusing Deep Learning & Physics-Based Models for Classification in Out-of-Distribution Scenarios

Deep Learning (DL) techniques have revolutionized the field of computer vision, over the last decades, enabling computers to perceive and understand their surrounding environment with unprecedented accuracy. Yet, these techniques frequently struggle with Out-of-Distribution (OOD) data, samples that differ significantly from the training distribution, leading to poor generalization performance. This chapter addresses this challenge by proposing fusing DL with prior scientific knowledge for the task of classification. After establishing the problem formulation, we present an analysis of the classifiers involved. A simple Convolutional Neural Network (CNN) classifier is employed to reproduce the issue of poor performance on OOD data. Based on the dynamics of the target object, a model-based classifier is proposed based on the Moving Horizon Estimation (MHE) framework. The outputs of the two classifiers are fused with the Dempster-Shafer (D-S) theory. Analysis and simulations explore the effectiveness of the proposed method highlighting its potential impact in the field of computer vision. Finally, an outlook to future work is proposed.

This chapter is based on the paper: A. Ilioudi, L. Gharavi, K. ter Burg, A. Dabiri, and B. De Schutter. "Deep learning and physics-based models fusion for classification in out-of-distribution scenarios". In: *Information Sciences* (2025). to be submitted.

Author contributions statement: A. Ilioudi co-conceptualized the proposed architecture with coauthors, developed the code for the implementation of the architecture, conducted the experiments, and wrote the manuscript.

7.1. Introduction

Deep neural networks have achieved impressive progress in various vision tasks, such as image classification, object detection, and instance segmentation. Despite these advancements, however, they still suffer significant performance degradation when tested in OOD scenarios. An approach to address this challenge is to integrate domain knowledge or physical principles with deep learning architectures. Physics-based models incorporate prior knowledge about system dynamics and physical laws, which are invariant to distribution shifts and remain valid even in OOD scenarios. By combining the strengths of data-driven methods with physics-based reasoning, robust systems can emerge that are capable of handling OOD data.

The aim of this chapter is to develop a classification framework that fuses appearance-based features extracted by deep neural networks with per-object dynamics captured through physics-based methods. By coupling these two methodologies, we aim to enhance OOD generalization and introduce causal reasoning into DL systems.

To capture the object dynamics, we employ the MHE framework [2, 220]. MHE provides an optimization-based approach to state estimation, using a sliding window of recent observations to improve accuracy. It has been shown to outperform traditional methods like the Kalman filter in terms of flexibility and accuracy, which makes it a suitable choice for video-based classification tasks [95].

Inspired by the approach in [267], we design a DL-based classifier that incorporates an uncertainty metric to account for the presence of OOD data samples. This uncertainty-aware classifier adjusts the confidence of its predictions, in order to adapt to samples with a distribution that differs from the training set. The classifier is then coupled with a physics-based classifier based on the MHE framework. This physics-based classifier leverages the temporal and dynamic properties of objects in video sequences, providing complementary insights that are typically overlooked by static, appearance-focused classifiers.

For decision fusion, we utilize the D-S evidence theory [298], which is known for its robustness in handling uncertainty and conflicting evidence. Compared to alternative methods, such as Bayesian fusion, D-S theory provides a more flexible framework for integrating information from multiple sources [197]. This fusion architecture allows us to handle uncertainty and combine evidence from the two sources, resulting in improved classification performance in OOD scenarios. More specifically, the contributions of this chapter are:

- A physics-based classifier is presented that is based on the MHE framework and that classifies target objects in videos by analyzing their dynamic behavior. This approach captures temporal patterns, providing a complementary perspective to appearance-based features.
- A fusion framework is developed that takes into account both an uncertainty-aware, DL-based classifier and a physics-based classifier.

This chapter is structured as follows. Section 7.2 presents an overview of the related work in this field. Section 7.3 defines the problem that is addressed in this

chapter. In Section 7.4 the proposed architecture is explained in detail. In Section 7.5 the rationale for generating an Ordinary Differential Equation (ODE)-driven dataset is detailed and the implementation and results are presented. Finally, Section 7.6 presents the conclusions and potential directions for future work.

7.2. RELATED WORK

The problem of poor performance on OOD generalization in DL has attracted significant attention over the last years, especially in the field of computer vision [310]. This limitation has led to the exploration of architectures that fuse various DL methods and employ uncertainty quantification techniques for improving robustness and interpretability.

A number of research works have focused on improving DL robustness to OOD data and uncertainty estimation. In [159] it is proposed to model the uncertainty of feature statistics during training to enhance generalization, assuming that feature statistics follow a multivariate Gaussian distribution. In [185] generative learning is combined with uncertainty to create a task-aware uncertainty measure for robotic perception, in order to improve generative approaches. A technique that estimates training data density in representation space to predict model uncertainty is introduced in [218]. In addition, [68] provided theoretical foundations for obtaining practical uncertainty estimates in DL, connecting Bayesian models with dropout and other stochastic regularization techniques. In [1] and [7] the application of uncertainty quantification to Artificial Intelligence (AI) is reviewed. Algorithms such as D-S theory, Monte Carlo simulation, and Bayesian inference are explored for handling uncertainty. Their findings show the importance of ensemble methods that fuse multiple techniques to address the uncertainty in real-world data. In [158] a fusion framework for multi-spectral pedestrian detection is proposed, integrating D-S theory to stabilize the detection process under noisy or corrupted data conditions. In [267] an uncertainty-aware fusion approach is proposed to fuse multi-sensor data that might suffer from a range of known and unknown degradation, using uncertainty measures to scale modality-specific outputs. While various methods have been proposed to enhance DL robustness to OOD data, they often face challenges such as limited generalization across diverse tasks, increased computational demands, and difficulties in handling conflicting information.

Building on these advances, the current work proposes a fusion framework that combines appearance-based features extracted by a CNN with model-based classifiers, based on prior knowledge. To the best of our knowledge, this is the first work to integrate DL techniques with dynamics knowledge in a fusion scheme. The results indicate that our method can be useful for incorporating domain knowledge into DL to improve the robustness of these algorithms.

7.3. PROBLEM STATEMENT

Given a video segment and an associated discrete-time state trajectory signal $\mathbf{z} = \{z(k)\}_{k=1}^{K}$, where k denotes the discrete time steps, of an object present in the

video segment $\mathbf{I} = \{I(k)\}_{k=1}^{K}$ consisting of consecutive image frames $I(k)$, $k = 1, \ldots, K$, the objective is to determine the most probable class category c^* from a predefined set of class categories $\mathbf{C} = \{c_1, \ldots, c_M\}$ to which the object most likely belongs.

In this chapter, we propose a classifier fusion technique that integrates two classifiers to combine information from the visual appearance captured in the video segment with domain-specific knowledge regarding the behavior of the object, as represented by its trajectory. More specifically, the first classifier is a CNN, trained in image frames for the classification task. The second classifier is a model-based one, concerning the dynamic behavior of the target objects with the aim of increasing the performance of the CNN classifier. Finally, the D-S framework is utilized to fuse the output of the two classifiers.

7.4. METHOD

In this section, the approach for the development of the appearance-based classifier is described first. Then the development of a model-based classifier is presented under the assumption that the dynamics of the target class categories objects are known. Finally, the fusion framework is discussed. The proposed architecture is depicted in Figure 7.1.

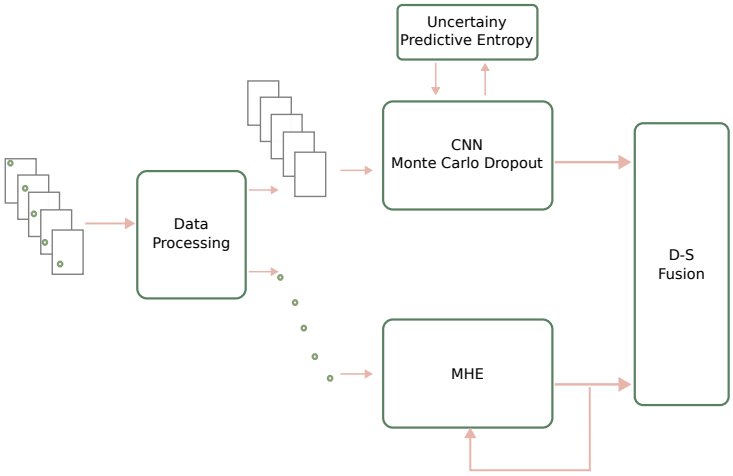


Figure 7.1: Proposed architecture.

7.4.1. APPEARANCE-BASED CLASSIFIER

For the appearance-based classifier, a CNN is employed. The CNN processes each image frame $I(k)$ through a series of layers, each applying specific operations to

extract image features. The input image $I(k)$ is first passed through a convolutional layer, which can be represented as $y(i+1) = f_{c_i}(y(i))$, where $y(i)$ denotes the input to the i-th layer, $y(i+1)$ is the output of that layer, and f_{c_i} represents the convolutional operation at layer i. Following the convolution, a Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity, to ensure the network's ability to model complex functions.

After the activation, a max-pooling operation is performed to reduce the spatial dimensions of the feature maps, downsampling the input. In addition, dropout is applied as a method to introduce stochasticity, by randomly setting a fraction of the input units to zero [250].

The processed feature maps are then passed through an adaptive average pooling layer, which ensures a consistent output size regardless of the input dimensions. This layer outputs feature maps of a fixed size, facilitating the transition to the fully connected layers. The output from the adaptive pooling layer is flattened into a one-dimensional vector and fed into a fully connected layer, which combines the extracted features to form a comprehensive representation. Another ReLU activation is applied before passing the result to the final fully connected layer, which produces the logits L_m corresponding to each class c_m.

To compute the probabilities p_{NN} for each class category c_m, the logits L_m are transformed using the softmax function:

$$p_{\text{NN}}(c = c_m | I(k)) = \text{softmax}(L_m) = \frac{\exp(L_m)}{\sum_{j=1}^{M} \exp(L_j)}. \tag{7.1}$$

This function converts the logits into a probability distribution over the M classes.

To ensure the robustness of the proposed classification framework, it is important to evaluate the confidence of the predictions, especially when encountering OOD samples. By quantifying uncertainty in the predictions, we can identify samples that deviate significantly from the training distribution and adjust the confidence of the classifier accordingly.

To achieve this, we use the predictive entropy measure [135]. More specifically, given a test sample $I(k)$ and training dataset D_{train}, the entropy of a distribution $H[c|I(k), D_{\text{train}}]$ captures the average amount of information contained in the predictive distribution and it is approximated by collecting outputs from T stochastic forward passes with different dropout samples through the network with Monte Carlo Dropout [68]:

$$\hat{H}[c|I(k), D_{\text{train}}] \approx -\sum_{m=1}^{M} \left(\frac{1}{T} \sum_{t=1}^{T} p_{\text{NN}}(c = c_m | I(k)) \log \frac{1}{T} \sum_{t=1}^{T} p_{\text{NN}}(c = c_m | I(k)) \right). \tag{7.2}$$

where $p_{\text{NN}}(c = c_m | I(k))$ denotes the predicted probability for class c_m given the input $I(k)$ at the t-th stochastic forward pass. In addition, to assess the degree to which a test sample deviates from the training distribution, the certainty level is calculated, which is less than 1 if a test sample is out-of-distribution and 1 if the sample is in-distribution [267]:

$$\delta(I(k)) = \frac{\mu_{\text{train}}}{\max(0, \mu_{\text{test}}(I(k)) - \mu_{\text{train}} - \sigma_{\text{train}}) + \mu_{\text{train}}}, \tag{7.3}$$

where μ_{train} is the training average of the entropy uncertainty metric aggregated across all images in the training set and σ_{train} is the standard deviation of the uncertainty metric scores. Moreover, $\mu_{\text{test}}(I(k))$ is the entropy uncertainty score for a test sample.

To calibrate the CNN classifier to reflect the uncertainty, we calculate the uncertainty-calibrated probability using equation 7.1:

$$\mathbb{P}_{\text{NN}}(c = c_m | I(k), \delta(I(k))) = \text{softmax}(L_m \delta(I(k))). \tag{7.4}$$

Thus, if a sample is far from the training distribution, δ will be a scalar less than 1. As a result, the distribution is softened and the CNN classifier is less confident in its prediction.

The probability measures in (7.4) are then used to formulate the evidence $P_{\text{NN}} : \{c_1, \dots, c_M\} \rightarrow [0,1]$ at time step k as follows:

$$P_{\text{NN}}(c_m, k) = \mathbb{P}_{\text{NN}}(c = c_m | I(k), \delta(I(k))). \tag{7.5}$$

7.4.2. MODEL-BASED CLASSIFIER

We assume that prior knowledge exists about the movement of the target object in the form of a state-space model as:

$$z(k+1) = f_m(z(k)), \tag{7.6}$$

where f_m corresponds to the dynamics of class c_m, if the object belongs to this class. Given the state measurements $\zeta(k)$, we assume a zero-mean Gaussian measurement noise $\nu(k) \sim \mathcal{N}(0, \sigma_\nu)$ such that $\zeta(k) = z(k) + \nu(k)$. As the name suggests, the multi-model MHE minimizes the expectation of estimation error for each class m over a moving horizon of N time steps as

$$E_m(k) = \min_{\hat{z}} \ \mathbb{E}\left(\sum_{i=0}^{N-1} \| \hat{z}(k-i|k) - z(k-i) \|_2 \right), \tag{7.7a}$$

$$\text{s.t.} \quad \hat{z}(k-i+1|k) = f_m(\hat{z}(k-i|k)), \quad i \in \{1, \dots, N\},$$

where $\hat{z}$ and E_m respectively denote the estimated state sequence and the expected estimation error for model m. We map the expected estimation errors for different models into the domain-knowledge probability of the class category c_m via the softmax operator as

$$\mathbb{P}_{\text{MB}}(c = c_m | z(k)) = \text{softmax}(E_m(k)). \tag{7.8}$$

The probability measures in (7.8) are then used to formulate the model-based evidence $P_{\text{MB}} : \{c_1, \dots, c_M\} \rightarrow [0,1]$ at time step k as follows:

$$P_{\text{MB}}(c_m, k) = \mathbb{P}_{\text{MB}}(c = c_m | z(k)). \tag{7.9}$$

7.4.3. EVIDENCE FUSION FOR CLASSIFICATION

The following fusion architecture combines predictions from multiple classifiers to take into account complementary information. Here, we present the case of fusing predictions from a CNN classifier (appearance-based) with a model-based one, that incorporates prior knowledge about the underlying dynamics of the target objects.

To perform this fusion, we employ the D-S framework [298], which allows the combination of evidence from different sources while accounting for possible conflict between their predictions. The fused evidence for the class c_m at time step k is given by:

$$P(c_m, k) = \frac{P_{\text{MB}}(c_m, k)\, P_{\text{NN}}(c_m, k)}{1 - \mathcal{K}(k)}, \tag{7.10}$$

where the degree of conflict between the evidence mass of the model-based and appearance-based classifiers at time step k is measured via the $\mathcal{K}$ function as:

$$\mathcal{K}(k) = \sum_{i \neq j} P_{\text{MB}}(c_i, k)\, P_{\text{NN}}\left(c_j, k\right), \tag{7.11}$$

where i and j represent different class indices. This measures the total disagreement between the outputs of the two classifiers. A higher conflict value indicates greater disagreement between the two classifiers. By incorporating $\mathcal{K}(k)$ in the denominator of the fusion rule, the D-S framework ensures that conflicting evidence is penalized, resulting in a final prediction that balances contributions from both classifiers.

7.5. CASE STUDY

In this section, we investigate the performance of the developed framework on a synthesized dataset driven by ODEs and then present and discuss the experimental results.

7.5.1. ODE-DRIVEN DATASET

To evaluate the proposed architecture, it is necessary to introduce an ODE-driven dataset in order to generate time-evolving trajectories of objects and their respective image samples which are necessary for the two classifiers to be trained and tuned. A mathematical model is developed for each object's class category, describing the 2D position and velocity of the object, i.e., free-fall or spring-damper systems. The Newtonian dynamics formulation has been considered to derive the mathematical representation of the dynamics of the target objects. In the Newtonian framework for a two-dimensional space, any system can be characterized by a position vector $\mathbf{r}$, such that

$$\mathbf{F} = m\, \ddot{\mathbf{r}}, \tag{7.12}$$

with m representing the mass of the object and $\mathbf{F}$ the total force exerted to it. Let $\mathbf{r} = (r_h, r_v)^\intercal$ denotes the vector of position coordinates of the target object with $r_h, r_v \in \mathbb{R}$. This vector consists of the 2D position of the target object in the horizontal and the vertical axis respectively. A state-space description of the system can then be derived using the states:

$$z = (z_1, z_2, z_3, z_4)^\mathsf{T} = (r_h, \dot{r}_h, r_v, \dot{r}_v)^\mathsf{T}. \tag{7.13}$$

In our case, we consider two classes, a free-falling system and a spring-damper system, which results in the following state space representations:

$$\text{Class 1:} \quad \begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = \alpha_1 u_1 \\ \dot{z}_3 = z_4 \\ \dot{z}_4 = \alpha_2 u_2 \end{cases} \tag{7.14a}$$

$$\text{Class 2:} \quad \begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = \kappa_1 z_1 + d_1 z_2 \\ \dot{z}_3 = z_4 \\ \dot{z}_4 = \kappa_2 z_1 + d_2 z_2 + \alpha_2' u_2 \end{cases} \tag{7.14b}$$

Some remarks on the above equations:

- Equations (7.14a) correspond to the dynamics of the system of class category 1 and equations (7.14b) correspond to the dynamics of the system of class category 2.
- The parameters $\alpha_i, \kappa_i, d_i, \alpha_2'$ $i = 1, 2$ represent constant parameters composed of stiffnesses, damping coefficients, masses, etc. and the inputs u_i correspond to acceleration (i.e. gravitational). Their numerical values are reported in Table 7.2 of Appendix 7.6.

Some trajectory samples are depicted in Figure 7.3.

7.5.2. EXPERIMENTAL RESULTS

To assess the effectiveness of the developed classification framework Python programming language is employed to perform a numerical simulation and generate the synthesized ODE-driven dataset.

To simulate the sample images and the trajectories, two shapes are drawn, randomly initialized, and driven by the ODEs in (7.14) on a rectangle canvas of size 640×380 pixels. The solutions of the ODEs are mapped to the canvas. Gaussian noise with a mean of $\mu = 0$ and a standard deviation of $\sigma = 0.3$ is added to the solution trajectories as measurement noise. Each video has 301 time steps and a rate of 30 frames per second. For the test set, Gaussian noise with a mean of $\mu = 0$ and a standard deviation of $\sigma = 0.568$, both relative to the normalized pixel intensity values in the range [0, 1], is added to the image samples. A sample visualization of the training set is shown in Figure 7.2 while the 2D trajectories extracted from the videos are depicted in Figure 7.3.

To evaluate the performance of the proposed framework, two illustrative examples are presented. Those examples concern the following cases:

- *Case 1:* The state estimation with MHE for system 1 (Figure 7.4).

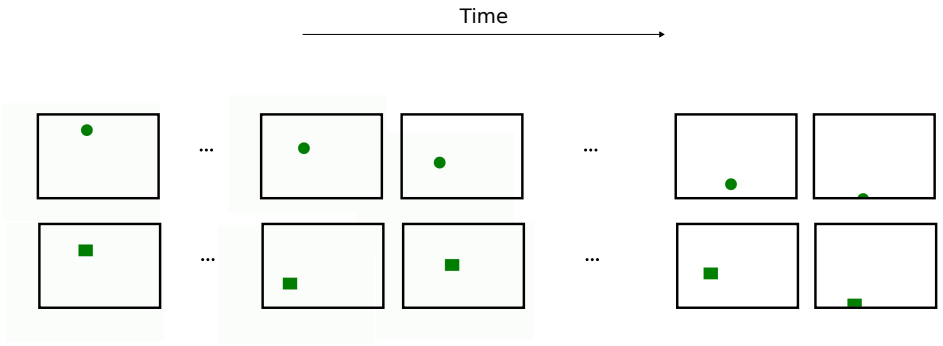
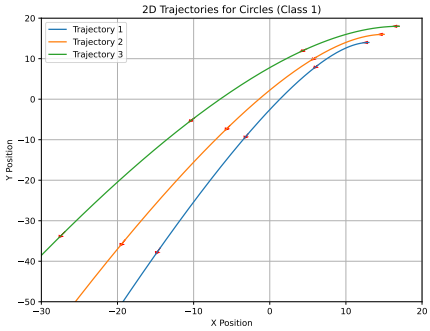
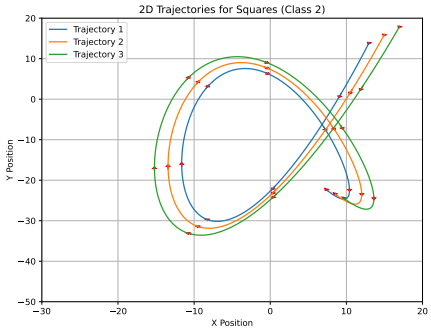


Figure 7.2: Illustration of the generated dataset showing the movement of two shapes (a circle and a square) over time, driven by the solutions of the ODEs. Each row represents a different class: the top row corresponds to the trajectory of a free-falling system (class 1), and the bottom row corresponds to a spring-damper system (class 2). The shapes are initialized randomly and move according to their respective dynamics. The frames $I(k)$ are sampled at different time steps to depict the evolution of the trajectories across time.

- *Case 2:* Evaluation of the performance of each classifier and the proposed fusion scheme (Figure 7.5 and Table 7.1).



(a) Class 1



(b) Class 2

Figure 7.3: Trajectories of the target objects in the generated dataset.

From the above figures and Table 7.1, it is observed that the combination of the two classifiers achieves a significant increase in classification accuracy. Figure 7.4 shows the state estimation of the MHE for class 1. Figure 7.5 presents the accuracy and the F1 score of the classifiers and their combination. The CNN classifier has a poor performance on the test set, which includes noisy samples. In that case, it can be seen that the fusion is very effective, as it integrates

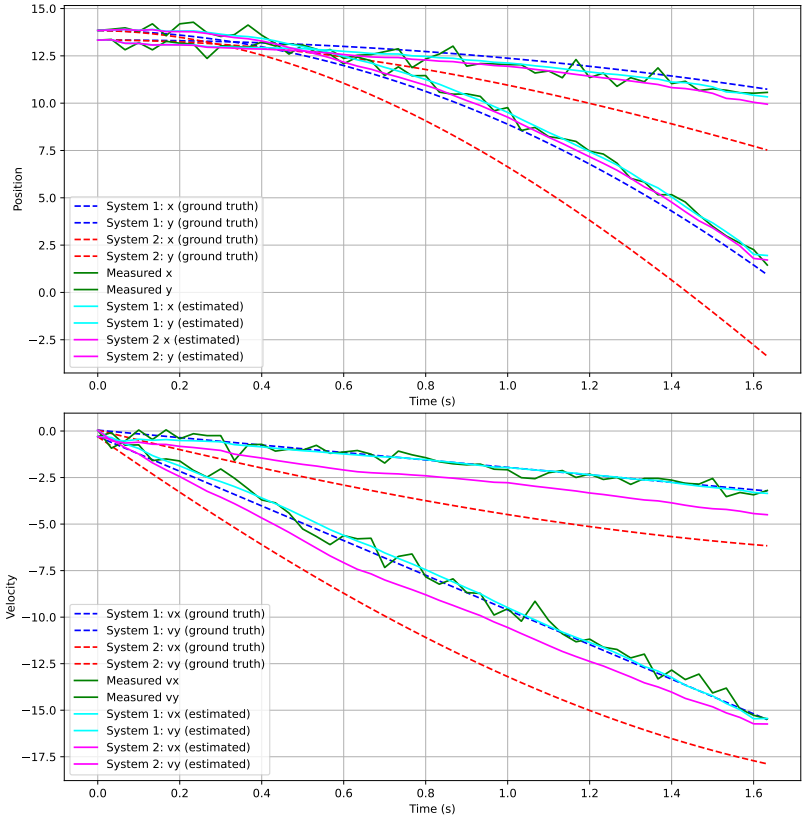


Figure 7.4: Comparison of estimated states for MHE with different dynamics constraints. The measured trajectories correspond to an object of Class 1.

Table 7.1: Comparison of classification performance for the classifiers and their combination.

Classifier Method	Accuracy	F1 score
CNN	60.2 %	21.7 %
MHE	72.9 %	77.0 %
DS Fusion	92.8 %	91.4%

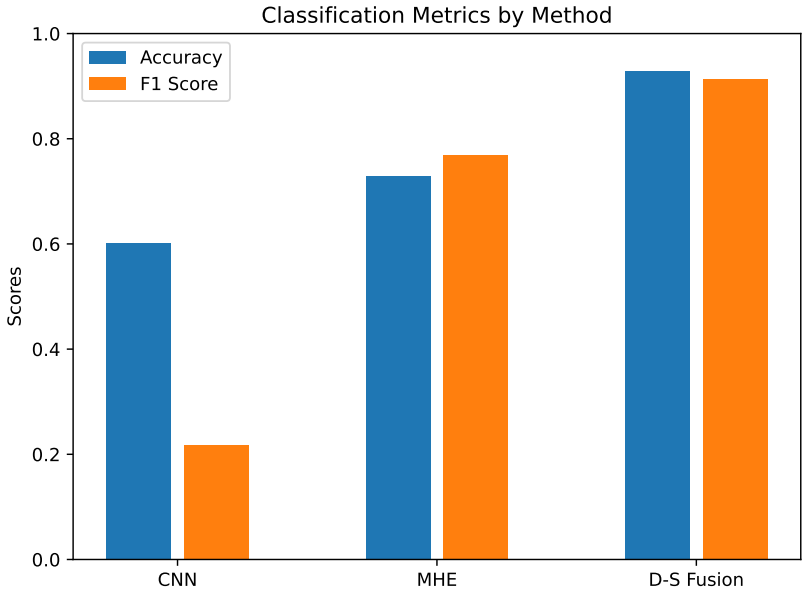


Figure 7.5: Performance comparison of each classifier and their D-S fusion.

the evidence of the two classifiers, taking into account when the output of the two classifiers are aligned or in conflict with each other. The details of the performance values for the classifiers and their combination are given in Table 7.1. The code for the developed architecture and the generated dataset are available at https://github.com/athinail/PINN_computer_vision.git.

7.6. CONCLUSIONS AND FUTURE WORK

This chapter has demonstrated the effectiveness of a fusion scheme that couples a CNN with a model-based classifier to address the poor performance of DL on OOD data. By generating an ODE-driven dataset with a distinct distribution shift between the training and test sets, the limitations of purely data-driven approaches have been highlighted. A CNN architecture has been employed to replicate the common issue of poor performance on test data, stressing the need for hybrid approaches. To achieve this, a model-based classifier has been proposed, which is based on the MHE framework. The outputs of the two classifiers are fused using D-S theory. The experimental results are promising, showing a significant increase in the F1-score metric of the classification task.

Future work should build upon these findings by developing more complex, ODE-driven datasets that further challenge the robustness of the proposed fusion scheme and its applicability to a wider range of DL architectures. This would allow for a more rigid validation of the approach on more sophisticated CNN architectures and real-world applications where distribution shifts are common. Expanding

the fusion scheme to include other types of DL and model-based techniques is another important research direction. Finally, validating this hybrid approach using real-world video data would be highly beneficial for assessing its performance in practical applications.

Appendix

7.A. System parameters

Parameter	Description	Value
$\alpha_1 = \alpha_2$	inverse of the body mass (Class 1)	1 kg^{-1}
α_2'	inverse of the body mass (Class 2)	2 kg^{-1}
κ_1	stiffness in the horizontal axis (divided by the mass)	0.4 N/m·kg
κ_2	stiffness in the vertical axis (divided by the mass)	0.4 N/m·kg
d_1	damping in the horizontal axis (divided by the mass)	0.2 Ns/m
d_2	damping in the vertical axis (divided by the mass)	0.2 Ns/m

Table 7.2: Description and Values of System Parameters

8

Domain-Informed Neural Networks: Detecting Underwater Moving Objects with Noisy Data

Underwater debris is a threatening environmental issue. While autonomous detection and collection of underwater waste are critical to address this challenge, they remain difficult due to the complexities of the underwater environment, including variable lighting conditions, reduced visibility, and the complex nature of water currents. This chapter presents a novel approach that incorporates domain knowledge to refine deep-learning-based underwater object detection techniques. More specifically, domain knowledge is represented by models in state space form that describe the motion of the target objects to assist the classification of objects. Moreover, optical flow is combined with a k-means clustering algorithm to extract the trajectory of the target objects from videos. These trajectories are subsequently fed into a neural network that is trained using knowledge distillation, enhanced with domain knowledge. The results of the experiments demonstrate that including domain knowledge within the object detection approach with neural networks provides numerous substantial advantages, including enhanced robustness against noisy and poorly labeled data, facilitation of semi-supervised learning, and consistent superiority in accuracy over the baseline scenario. An increase in accuracy of up to 10% is observed in the classification. Additionally, combining the domain knowledge with a neural network significantly increases the computational speed of object detection compared to using the standalone domain knowledge module.

This chapter is based on the paper: M. D'heer, A. Ilioudi, B. De Schutter, A. Dabiri, and A. Jamshidnejad. "Domain-informed neural networks for detecting underwater moving objects with noisy datasets". In: *Knowledge-Based Systems* (2025). to be submitted.

Author contributions statement: A. Ilioudi conceptualized the proposed architecture, contributed to

8.1. Introduction

While the production of disposable products has significantly increased in recent decades, the ability to sustainably manage the waste has not kept pace with the rising production. Especially in the oceans, it is estimated that there are between 50 and 75 trillion pieces of plastic, and this number is continuously rising [58]. The removal of underwater debris is highly resource and time-intensive, making autonomous methods the most feasible and scalable solution. The first step in enabling these autonomous approaches is to achieve automated debris detection from captured images. The underwater environment poses various challenges compared to other object detection problems, such as low contrast, uneven illumination, complex backgrounds, and high noise caused by water turbulence [43, 64, 125, 297].

Relying solely on frame-by-frame object detectors in video analysis results in the loss of critical temporal information. To address this limitation, video-based object detectors that incorporate temporal data are essential. However, training these detectors effectively is a significant challenge due to the complexity of modeling the temporal dependencies [211]. Moreover, acquiring datasets of sufficient size and diversity for training such models is very difficult, if not impossible. Typically, data is gathered from a specific environment and is used for training; however, these models often fail to generalize well to other environments. This limitation requires retraining and fine-tuning of the model for each new deployment.

An alternative approach involves the integration of domain-specific knowledge into the learning process, known as informed machine learning [280]. The primary advantage of this approach is the assurance that the underlying physics or domain principles are embedded within the data-based architecture, reducing the uncertainty that arises when relying solely on data-driven methods.

This chapter proposes a pipeline for integrating domain-specific knowledge, relevant to underwater object movement dynamics, with a neural network for the detection of underwater moving objects. More specifically, our contribution consists of the development of a novel domain-informed neural network architecture that addresses the challenges of underwater object detection with limited and poor-quality data. To evaluate our approach, we also propose a new dataset for benchmarking domain-informed neural networks, i.e., a synthetic Ordinary Differential Equation (ODE)-driven dataset, inspired by real-world benchmarks [52]. The proposed architecture is trained in both a fully supervised and a semi-supervised learning setting and the experiments show an increase in classification accuracy of up to 10%.

The resulting architecture autonomously detects living marine beings from objects to ensure the safe removal of the trash without harming live beings. Given the scope of this research, the chapter is limited to 2D motion and two primary class categories, including trash and fish. The trash class consists of two objects: a plastic bottle and a plastic beer holder. The fish class comprises a single type of fish. The proposed approaches can be generalized and expanded to 3D and more classes and different applications, including various types of trash, other marine life, or even other applications that use different types of domain knowledge.

the code development, and reviewed the manuscript.

This chapter is organized into seven sections: Section 8.2 discusses the related work. The proposed architecture is described in Section 8.3. In Section 8.4 the simulator setup, the dataset generation, and the data preprocessing module are described. Section 8.5 describes the training process of the neural network. The results are discussed in Section 8.6. The conclusion and topics for future research are given in Section 8.7.

8.2. RELATED WORK

Domain knowledge incorporation methods can be broadly categorized into logical and numerical constraints, and they can be applied across different domains [44]. Integration of domain knowledge in neural networks is known to improve the performance, consistency with the principles of physics, and generalizability of neural networks [6, 61, 217]. The domain knowledge can be integrated within a neural network via the input data, the loss function, knowledge distillation, the architecture, or the output data [44]. For example, in [120] prior domain knowledge regarding the structure of the human body is used in a physics-based deep reinforcement learning approach in order to estimate the 3D human position.

In [238] rules are integrated into the training of a neural network for regression to ensure that the inference output is consistent with physics laws. Similarly, [34] introduces a rule encoder and a text encoder to create embeddings for hateful content and the logical rules underlying it. Through contrastive learning, the proposed architecture employs a semantic similarity objective, pairing instances of hateful content with clusters of rule instances that define the content's governing principles. In [263] self-organizing neural networks incorporate symbolic domain knowledge translated into numeric patterns, enhancing reinforcement learning in complex domains. In [252] an approach of supervising networks using constraints derived from domain knowledge, such as physics laws, instead of labeled examples is presented, in order to significantly reduce the need for labeled training data in computer vision. Finally, using knowledge distillation [106], a method that involves transferring information from a (typically large) teacher neural network to a (usually small) student neural network, Hu et al. [114] have shown the efficacy of combining deep neural networks with structured logical rules. Their work will serve as a foundational basis for this chapter.

8.2.1. DATASETS AND SIMULATORS

Data is paramount in any machine learning problem. In fact, the quality of the dataset that is used in the training and validation of a machine learning method has a major effect on the performance of the resulting model [30, 88]. As this chapter focuses on dynamic underwater environments, the machine learning-based classification method requires a sequentially annotated dataset including both plastic and fish. The most known underwater datasets are Deep-sea Debris Database [124], Trashcan [109], and TrashICRA19 [65]. However, these datasets provide annotations at the frame level, instead of the video level, and lack unique identifiers to track the moving objects. This makes none of the publicly available datasets usable for this

research. In addition, given the scarcity of real-world video object detection datasets, synthetic datasets are considered. The most widely employed, open-source visually and physically accurate simulators, are the Gazebo-based simulator [179] integrated into Robot Operating System (ROS) [212] and a Unity-based simulator [260]. Gazebo offers a robust environment with highly accurate physics and rendering capabilities, making it suitable for testing complex robot interactions in dynamic settings. Unity-based simulators, on the other hand, provide advanced graphics capabilities of the Unity game engine to create visually rich environments. This is particularly beneficial for applications requiring realistic visual representation, e.g., in computer vision. Each platform offers different advantages: Gazebo features high physical accuracy and ROS integration, while Unity-based simulators provide superior visual fidelity and a vast asset library.

8.3. Proposed framework

This section analyzes the mathematical model representing the domain knowledge and its integration with a neural network.

8.3.1. Mathematical models for incorporating domain-specific knowledge

A mathematical model is developed per class of objects, where the modeling focuses on underwater movement, while partial submersion is outside the scope of this chapter. The mathematical model represents the 2D position and velocity of each object class through a set of nonlinear differential equations. Both fish and plastic objects experience similar hydrodynamic forces, namely gravity, buoyancy, and drag. However, in the case of the fish, there is an additional propulsion force that models the thrust generated by the tail [156]. This force is introduced as a sinusoidal input acting on the horizontal axis, while the fish regulate their position on the vertical axis using a swim bladder, an internal gas-filled organ that modulates buoyancy. To unify the representation, we use the same system of equations for all object classes and introduce a switch coefficient $\sigma \in \{0, 1\}$ to account for the fish propulsion force. Denoting the location of an object on the vertical and the horizontal axis as p_{v} and p_{h} respectively, the state vector is defined as $\begin{bmatrix} s_1 & s_2 & s_3 & s_4 \end{bmatrix}^T$ with $s_1 = p_{\mathrm{v}}^o$, $s_2 = \dot{p}_{\mathrm{v}}^o$, $s_3 = p_{\mathrm{h}}^o$ and $s_4 = \dot{p}_{\mathrm{h}}^o$ where $o \in \{\text{fish, bottle, holder}\}$ (the differences between the plastic and the fish are indicated in bold). The nonlinear system of equations of

motion is given by:

$$
\begin{aligned}
\dot{s}_1 &= s_2 \\
\dot{s}_2 &= \frac{1}{m^o}\left(\overbrace{-\overbrace{m\cdot g}^{\text{Gravity}}+\overbrace{\rho^{\text{water}}\cdot g\cdot V^o}^{\text{Buoyancy}}}^{}-\underbrace{\frac{1}{2}\text{sgn}(s_2)\cdot c_{\text{v}}^{\text{d},o}\cdot\rho^{\text{water}}\cdot A_{\text{v}}^o\cdot {s_2}^2}_{\text{Drag}}\right) \\
\dot{s}_3 &= s_4 \\
\dot{s}_4 &= \frac{1}{m^o}\left(\boldsymbol{\sigma}\cdot\left(\overbrace{\boldsymbol{F}^{\text{f}_\text{h}}\cdot\boldsymbol{u_2}+\boldsymbol{F}^{\text{f}_\text{h}}}^{\text{Fish force}}\right)-\frac{1}{2}\cdot\text{sgn}(s_4-v_{\text{h}}^{\text{current}})\cdot\frac{1}{2}\cdot c_{\text{h}}^{\text{d},o}\cdot\rho^{\text{water}}\cdot A_{\text{v}}^o\cdot(s_4-v_{\text{h}}^{\text{current}})^2\right) \\
\boldsymbol{u_2} &= \sin(\boldsymbol{t}\cdot\boldsymbol{f}^{\text{tail}}).
\end{aligned}
\tag{8.1}
$$

Based on the condition of the switching coefficient, this system corresponds to the movement of the plastic (for $\sigma = 0$) and the fish (for $\sigma = 1$). The static parameters include the gravitational acceleration g, the mass of the object m^o, the density of water ρ^{water}, the sea current's speed $v_{\text{h}}^{\text{current}}$, the drag coefficient for object o on the vertical axis $c_{\text{v}}^{\text{d},o}$, and the drag coefficient for object o on the horizontal axis $c_{\text{h}}^{\text{d},o}$. The other parameters are volume V^o of the object and the surface areas A_{v}^o and A_{h}^o with $o \in \{\text{fish, bottle, holder}\}$, where A_{v}^o represents the cross-sectional area of the object perpendicular to the vertical axis, A_{h}^o represents the cross-sectional area of the object perpendicular to the horizontal axis, and t represents continuous time. Finally, for the fish mode, $\boldsymbol{F}^{\text{f}_\text{h}}$ corresponds to the "fish force" and a tail frequency $\boldsymbol{f}^{\text{tail}}$.

8.3.2. DOMAIN KNOWLEDGE INTEGRATION

To incorporate domain-specific knowledge into the neural network training process, we employ the concept of knowledge distillation [114]. Knowledge distillation is typically applied to replace a large-sized neural network (teacher network) with a smaller one (student network) that aims to match the teacher's performance. In this chapter, instead of a large teacher network, we use the mathematical model introduced in (8.1) as the "teacher". This provides a form of regularization during training, guiding the student network towards physically consistent solutions.

In a standard classification problem, the student network outputs a set of raw scores, known as logits, before applying the final softmax activation. Logits are real-valued numbers (one for each class) that represent the raw likelihoods of each class. Applying the softmax function to the logits normalizes them into a probability distribution over the classes.

During the training process, the student not only learns from the data but also from the teacher. It does so by calculating the loss consisting of a weighted combination of the Cross Entropy (CE) loss, which measures how well the student's predicted probability distribution matches the true distribution from the labeled data, and the Kullback-Leibler (KL) divergence loss, which encourages the student to align with the softened probability distribution provided by the teacher. These losses

are, respectively, defined as:

$$L^{\text{ce}}\left(P^* \mid P\right) = -\sum_{i=0}^{2} P_i^* \cdot \log P_i \tag{8.2}$$

$$L^{\text{kl}}(P^{\text{soft}} \mid Q^{\text{soft}}) = \sum_{i=0}^{2} P_i^{\text{soft}} \log \frac{P_i^{\text{soft}}}{Q_i^{\text{soft}}}, \tag{8.3}$$

where P^* is the true class probability distribution from the data labels, P is the probability distribution predicted by the student neural network, and Q^{soft} is the predicted probability distribution determined by the teacher network. Moreover, P^{soft} is the softened predicted probability of the student network and it is obtained by dividing the logits by a temperature parameter T, which controls the softness of the probability distribution. Finally, P_i^*, P_i, P_i^{soft}, and Q_i^{soft} specify the probabilities of the i-th class in the true distribution, the predicted distribution, the softened predicted distribution, and the softened target distribution, respectively. The predicted probability is derived from the raw logits, after applying the softmax function:

$$P_i = P(c = c_i | h(0) \dots h(k)) = \text{softmax}(l_i) = \frac{\exp(l_i)}{\sum_{j=1}^{m} \exp(l_j)}, \tag{8.4}$$

with $[l_i, l_2, ..., l_m]$ the vector of the logits, m the number of classes, and $h(1) \dots h(k)$ the input trajectory to the network. For the softened probabilities, the logits are divided by the temperature parameter and then the softmax operator is applied:

$$P_i^{\text{soft}} = P(c = c_i | h(0) \dots h(k)) = \text{softmax}(l_i) = \frac{\exp(\frac{l_i}{T})}{\sum_{j=1}^{m} \exp(\frac{l_i}{T})}, \tag{8.5}$$

8

In this chapter, we propose to deploy the teacher-student architecture, but instead of a large-size neural network, we represent the teacher by the domain knowledge module, which employs the mathematical model (8.1) to generate the trajectories of movement per class. Each of these trajectories is then compared to the measured trajectory and based on the respective Mean Squared Error (MSE), their similarity is assessed. The class confidence for the domain knowledge module is calculated with the softmax operator, similarly to (8.5), but instead of the logits, the MSE values are used.

The total loss function becomes a weighted sum of the CE loss L^{ce} and the KL divergence loss L^{kl} given by:

$$L^{\text{tot}}(P^*, P, P^{\text{soft}}, Q^{\text{soft}}) = (1 - \alpha)\, L^{\text{ce}}(P^* \mid P) + \alpha\, T^2 L^{\text{kl}}(P^{\text{soft}} \mid Q^{\text{soft}}), \tag{8.6}$$

with α the parameter that determines the impact of both the CE loss and the KL divergence loss on the total loss. The KL divergence loss is multiplied by T^2 to ensure that the gradients remain appropriately scaled, maintaining the balance between the contributions of the CE and KL divergence losses [106]. Both T and α are hyperparameters tuned to achieve the highest accuracy.

8.4. ODE-DRIVEN DATASET

This section provides a comprehensive analysis of the proposed approach regarding the generation and curation of the dataset for training and validation of the proposed architecture. We also discuss the preprocessing techniques for motion trajectory extraction.

8.4.1. GENERATION AND CURATION OF DATA

Since no open-source dataset is available that can be used to extract the trajectories of underwater objects, custom data via a simulator is generated. The existing simulators that focus on Remotely Operated Vehicles (ROVs) lack the option to model plastic and fish in the underwater environment. Thus, custom modifications are needed to provide the desired underwater simulation environment.

SIMULATION MODEL WITH UNITY

In this chapter the Unity Game Engine simulator (briefly called Unity) [273] is chosen for its ability to accurately capture the underlying physics of underwater motion, while also offering realistic visuals that closely mimic real-world underwater environments. The overall architecture of the simulator is represented in Figure 8.1 and the ID's of the block are explained next. First, Unity simulates an ROV with a stationary camera, focusing on underwater object avoidance. A Python module interacts with Unity in the baseline simulator and gathers data. Given that this chapter is focused on the detection of moving objects, rather than on the movements of the ROV itself, the baseline simulator has been extended with a custom C# module for data gathering. Since C# provides direct Unity integration, it is a more suited option than Python. The C# module autonomously produces a dataset, where the desired number of trajectories for the moving objects is provided as input to the module. The scene of the scenario and the various plugins (including Aura 2 and custom scripts) are initialized as soon as the Unity editor is launched (block 1 in Figure 8.1). This then triggers the data-gathering script, which generates the required number of trajectories in a loop with several iterations, where during each iteration, various steps are executed (block 2 in Figure 8.1).

A custom-built module called the variable tracker (block 3.a in Figure 8.1), tracks the states of the moving objects. Two plastic objects i.e. a plastic bottle, a plastic beer holder, and a type of fish are considered. The position, velocity, and acceleration in the horizontal and vertical directions of an object are tracked. For the environment, the fog densities, the bubbles, the particle emission, and the current velocity are tracked.

The data generation procedure ensures a diverse dataset, by iteratively generating instances with varying parameters for the scene (such as the underwater fog strength, the particle emission rate, the intensity, the position of the light, and the direction and strength of the water current) and for the moving objects (these parameters include the mass, volume, initial position, and orientation), but keeping the water density, ground textures, and dimensions of the scene constant (block 3.b in Figure 8.1). Each iteration is characterized by the initialization of the variable

tracker, the generation of a random scene, and by spawning one randomly chosen object from the input list of objects (which includes fish, plastic bottle, or plastic beer holder) (block 3.c in Figure 8.1).

The underwater fog strength is a parameter that determines the water transparency, where a large value for this parameter indicates low transparency, which results in low visibility and affects the maximum distance that an object can be placed from the camera of the ROV, before becoming undetectable for it. The particle emission rate introduces particles (called the marine snow) in the scene that impact the clarity of the images taken by the ROV. The bubble emission rate determines the amount of bubbles (e.g. from animals or air escaping from the ROV) that are spawned in the simulation environment every second. Including these bubbles in the simulation enhances the visual fidelity of the scene. Similarly, the intensity and position of the light significantly impact the visual fidelity of the scenes, where by avoiding extreme lighting conditions more realistic scenarios are simulated. Finally, the direction and strength of the water current influence the direction and velocity of the movement of the objects. The ranges of all these parameters have been carefully chosen for the simulations in our case study to reflect realistic underwater conditions. Finally, by adding ocean sand and rock formations at the bottom of the scenes, the visual fidelity of the simulator has been further improved (see Figure 8.2).

Object spawning

The last step in the initialization is the spawning of either a plastic bottle, a plastic six-pack holder, or a fish, where these have been illustrated in, respectively, Figure 8.3a, Figure 8.3b, and Figure 8.3c. These objects are not only characterized by their mesh but also by their various properties, including the mass m^o, the volume V^o, the surface areas A_x^o, A_y^o, A_z^o and the drag coefficients $c_x^{\mathrm{d},o}$, $c_y^{\mathrm{d},o}$, $c_z^{\mathrm{d},o}$ in the 3 dimensions, and the density ρ^o, with $o \in \{\text{fish, bottle, holder}\}$. Moreover, the fish force F_z^{fish} is considered, which consists of a sinusoidal force used to simulate the thrust generated by the tail of the fish while swimming. In each iteration of our simulations, the mass and the volume of the plastic bottles, plastic six-pack holders, and fish are varied within a range of ±10% from their default values. This variation accounts for natural differences in object properties or biological diversity, in order to ensure realistic conditions for the simulation. During and across the iterations, the camera of the ROV is kept stationary at the position $[0, 0.5, 0]^\top$. The camera of the ROV has a vertical field of view of 60 degrees and uses perspective projection. While the simulator is capable of generating 3D trajectories, in order to reduce the complexity, the movements of the object are restricted along the depth axis (negative x-axis in Figure 8.2). Consequently, the object moves within a 2D plane relative to the camera.

An object is randomly selected for the iteration (block 4 in Figure 8.1). The upper bound of the position where the object is spawned across the horizontal axis is determined by the fog density ρ^{fog}. Given that the relationship between the fog density and the maximum distance from the camera that an object is still detectable for the camera is non-linear, the following polynomial regression is used

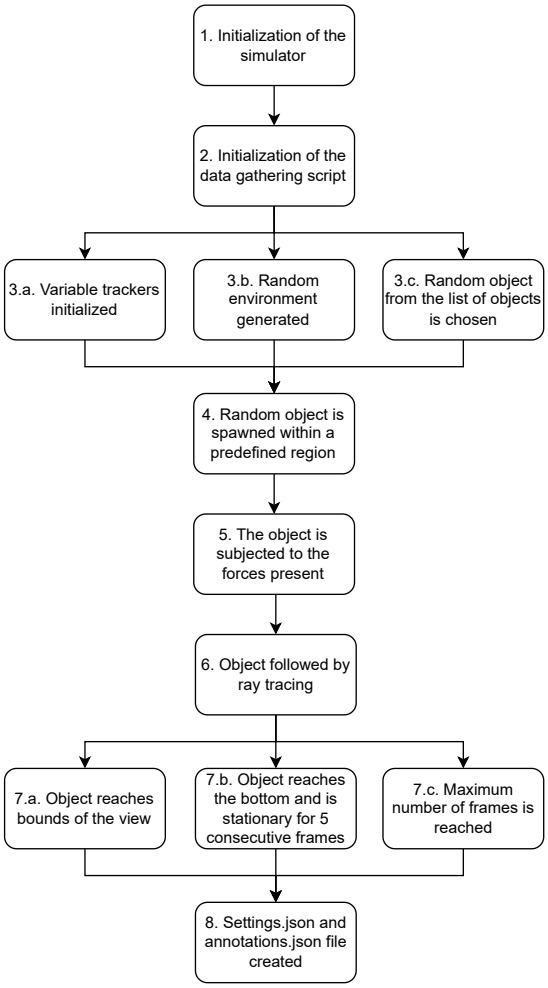


Figure 8.1: High-level architecture of the steps within the Unity simulator for generating the underwater environment for data collection and for running our experiments

to determine the bound for the depth coordinate of an object:

$$x^{\text{obj,max}} = 4 \cdot 10^{-6} \cdot \left(\rho^{\text{fog}}\right)^3 - 0.0011 \cdot \left(\rho^{\text{fog}}\right)^2 + 0.1048 \cdot \rho^{\text{fog}} - 5.0222 \tag{8.7}$$

After determining the bounds of the horizontal coordinate (z-axis), the bounds of the



Figure 8.2: Default scene of the underwater environment generated by Unity

vertical (y-axis) and depth (x-axis) coordinates are calibrated to ensure alignment with the camera's field of view. For this, the method `ViewportToWorldPoint` [274] from Unity is used, which maps points from the camera's viewport space to the world coordinate system. This ensures that the spawning bounds are consistent with what is visible to the camera in the simulation. Additionally, the direction of the water current is considered to restrict the spawning location to either the left-hand or right-hand side of the horizontal (z-axis) plane. This consideration, along with the implementation of buffer zones, constitutes the measures taken to prevent the object from being spawned at the periphery of the visible area and to ensure that the object does not immediately exit the field of view of the ROV.

The object spawns in the scene after the initialization steps are all complete. The Unity engine operates on a continuous cycle wherein the object states are updated, the inputs are processed, the calculations are performed, and the graphics are rendered. Each C# class within the simulation has two default methods: `Update()` and `FixedUpdate()`. The `Update()` method is invoked at every frame of the simulation video, but the frame rate varies with the performance of the hardware. To ensure that the calculations are executed consistently, the `FixedUpdate()` method is employed.

Finally, an iteration is terminated when one of the following three exit criteria is met:

1. The object has moved out of the view of the camera (block 7.a in Figure 8.1).

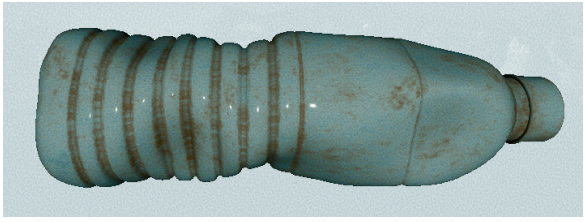
2. The velocity in the horizontal (z) and vertical (y) axis of the object has dropped under the threshold of moving at least 0.02 m between any two frames (block 7.b in Figure 8.1).

3. The maximum number of frames is reached (block 7.c in Figure 8.1)

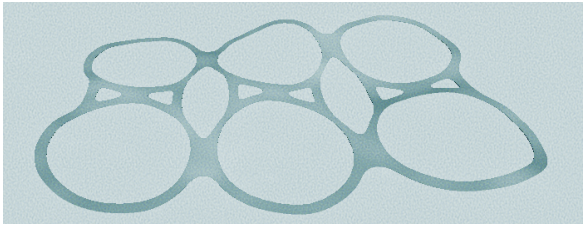
OUTPUT FILES

Upon completion of each iteration, the following two files are generated: *settings.json* and *annotations.json* (block 8 in Figure 8.1). The file *settings.json* contains a comprehensive record of the settings that have been employed in the simulation for the completed iteration. The file *annotations.json* is generated using a custom-built annotation module that casts rays across a designated area towards the center of gravity of the object. Each ray outputs a boolean that is true if the object is intersected by the ray and is false otherwise. An array is then created from all the rays that intersect the object and is then translated into pixel locations, facilitating the automated generation of the bounding boxes (block 6 in Figure 8.1). A sketch of this approach can be found in Figure 8.4.

All the aforementioned steps are repeated per iteration until the desired number of trajectories of movement are generated.



(a) Plastic bottle 3D model [227]



(b) Plastic beer holder 3D model



(c) Fish 3D model [63]

Figure 8.3: Objects used in the simulator

8.4.2. DATA PREPROCESSING

As explained earlier, the simulator produces a set of trajectories for the movement of the objects, where each trajectory corresponds to a set of sequential images,

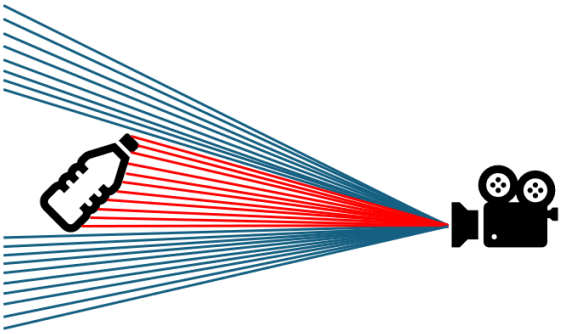


Figure 8.4: Sketch of bounding box creation module: The camera projects 200 rays in both the horizontal (z-axis) and vertical (y-axis) directions, where the sketch shows a 2D case.

a configuration file, and an annotation file. Although both the bounding boxes and the 3D positions of an object are always known to the simulator, in the real world, only the images with metadata are present. Consequently, in this subsection, we discuss how the patterns of movement captured from sequential images are leveraged to accurately infer and extract the real-world positions of the objects from the simulated data.

MOTION EXTRACTION

The motion extraction from sequential images is conducted through optical flow, specifically utilizing the Farneback method [57], which is widely used in such applications and is known for its high accuracy and robustness. Optical flow is a technique used to estimate the apparent motion of objects between two sequential frames, based on changes in pixel intensity patterns. This process is implemented using the OpenCV library [24]. The input consists of two sequential images that are likely to include a moving object, and the output is a velocity magnitude plot for the object.

In this work, the camera is fixed in position, thus all detected motion corresponds to moving objects in the scene rather than camera movement. However, in future work, we aim to extend these techniques to scenarios involving moving cameras. Suppose that image 1 corresponds to time instant t_1 with the object positioned at $(y_{t_1}^{\text{pix}}, z_{t_1}^{\text{pix}})$ expressed in pixel coordinates, and image 2 corresponds to time instant t_2 with the object positioned at $(y_{t_2}^{\text{pix}}, z_{t_2}^{\text{pix}})$ in pixel coordinates. Each image consists of multiple items with distinct optical flows, including the object of interest, bubbles, particles, waves, and present turbulence. If the actual velocity of the object through the water exceeds the velocity of, e.g. the bubbles or the waves, then the optical flow values for the object will be larger than those for the bubbles or the waves. Note that the bubbles, partial waves, or any turbulence will be referred to as the "surroundings" of the object. Given the average frame rate of 10 fps, a small, linear

motion for the object in between any two consecutive frames is assumed. We also assume that the velocity of the object exceeds that of the surroundings, thus the largest optical flow will be attributed to the object. Using these assumptions, the position of the moving object in pixel locations at time instant $\bar{t} = \frac{t_2 - t_1}{2}$ is approximated by:

$$(y_{\bar{t}}^{\text{pix}}, z_{\bar{t}}^{\text{pix}}) = \left(\frac{y_{t_2}^{\text{pix}} - y_{t_1}^{\text{pix}}}{2}, \frac{z_{t_2}^{\text{pix}} - z_{t_1}^{\text{pix}}}{2} \right) \tag{8.8}$$

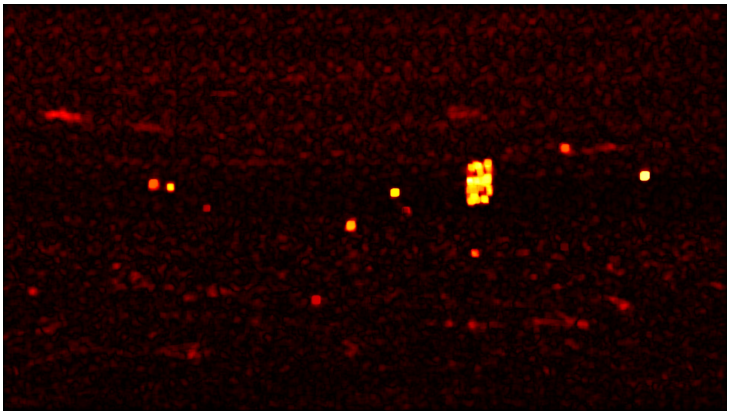


Figure 8.5: Output mask of the Farneback optical flow applied to two sequential underwater images of a moving plastic beer holder: The more intense the color of the pixel, the larger the optical flow is for that location.

From Figure 8.5, the object silhouette is not uniformly detected, and noise is present in the mask due to the dynamic surroundings. To accurately locate the centroid of the object, a k-means clustering algorithm [195] is applied. Firstly, the clustering algorithm is applied to the magnitude of the optical flow values. The algorithm is designed to extract three magnitude-based clusters, from which the background (any pixels not belonging to the object) is identified if it covers more than 20% of the image. This can be either one cluster or two clusters whenever a high level of noise is present. Subsequently, the resulting cluster or clusters undergo a second iteration of k-means clustering. This time the emphasis is on the spatial positioning of the highest optical flow values. The objective of this phase is to determine those segments of the previously identified cluster(s) that correspond to the actual object, given that the magnitude clustering is also susceptible to noise interference, due to optical flow produced by the surroundings. Ultimately, this process isolates a maximum of six clusters (the maximum six number of clusters occurs in case three magnitude and three color clusters are identified), and the centroids are calculated per cluster. These centroids are then assessed for their proximity to each other and the clusters with centroids closer than a given threshold are merged. Finally, the cluster of the most substantial size is presumed to represent the object of interest. The center of the resulting cluster is then extracted and

expressed in the local reference frame of the image.

TRANSFORMING PIXEL COORDINATES INTO GLOBAL ONES

The motion extraction module expresses the estimated location of the moving object in pixel coordinates. In order to use differential equations corresponding to the domain knowledge, these pixel coordinates should be transformed from a local 2D reference frame into a 3D global real-world frame of reference. The simulation uses the projection view [178] to generate the footage. The projection view formula to convert the y coordinate of a local reference frame in pixels to a global reference frame in meters is given by

$$y^{\text{obj}} = \left(\left(\frac{y^{\text{pix}} - y^{\text{center}}}{f} \right) d^{\text{cam}} - y^{\text{cam}} \right), \tag{8.9}$$

where y^{pix} is the given y-coordinate of the moving object in pixels and y^{center} is the y-coordinate of the center of the image in pixels, which equals 540 pixels, as the height of the image is 1080 pixels. Moreover, d^{cam} is the distance of the object from the camera in meters that is assumed to be known [99]. Additionally, y^{cam} is the offset of the coordinates of the camera that equals 0.5 m in this setup, and f is the focal length. The simulator does not provide the focal value, but this value is calculated using the following formula, taken from [108]:

$$f = \frac{h^{\text{pix}}}{2 \tan\left(\frac{\text{fov}^{\text{v}}}{2} \right)}. \tag{8.10}$$

The formula uses the resolution height h^{pix} and the vertical field of view fov^{v}, which equals 60 degrees in our setup.

These results, settings, and annotations are consolidated into a single .json file, which eliminates the need for interfacing with various files. In addition to the simulation and the optical flow-derived trajectories, to mimic the sensor noise that is typically present in real-world data, additive Gaussian noise with mean 0 and a standard deviation of 0.4 is included in the ideal trajectories. The unmodified trajectories are referred to as “ideal” trajectories.

In summary, the `parsed_data.json` file is compiled utilizing the data preprocessing module to combine all the data into a single file.

8.5. NEURAL NETWORK AND TRAINING

The neural network can be trained both with and without the knowledge distillation approach, by switching between learning from both data and domain knowledge to learning only from data.

The goal of the neural network is to classify the extracted trajectory of the target objects. A simple feed-forward neural network is chosen with one input layer, two hidden layers, and an output layer. The hidden layers are fully connected and each is followed by a Rectified Linear Unit (ReLU) activation function, which introduces

non-linearity into the neural network and facilitates the ability of the neural network to learn more complex functions. The input layer is configured with a fixed size of 1000 neurons, and the first and second hidden layers include respectively, 128 and 64 neurons. Reducing the number of neurons across the layers is to distill the most salient features of the data as it progresses throughout the neural network. Finally, the output layer maps the last hidden layer to the different classes of objects (see Figure 8.6).

A custom dataset class is used to parse and iterate through the data to ensure a streamlined training process. The training data for the neural network consists of an input trajectory transformed into a tensor of the format $[y^o, z^o, v_y^o, v_z^o, 0, 0, ...]$. Zero padding is utilized to meet the fixed input requirement with variable trajectory lengths. During training without domain knowledge, α (see (8.6)) is equal to zero and training relies solely on the CE loss function. The Adam optimizer [139] is used to solve the optimization problem (8.6).

When domain knowledge is included in the training procedure, both the trajectory of the detected object and the settings used to generate the trajectory (such as the distance of the object from the camera) are fed into the domain knowledge module, which then calculates the dynamics of motion based on the mathematical model for both classes, i.e. plastic and fish. The estimated dynamics are then compared with the input trajectory, and the logits are calculated and undergo a softening process, being divided by a temperature parameter. Utilizing these adjusted logits, the difference between the probability distributions generated by the neural network and the mathematical model is quantified using the KL divergence loss. The total loss is then back-propagated through the model in order to find the gradients and to optimize the weights of the neural network.

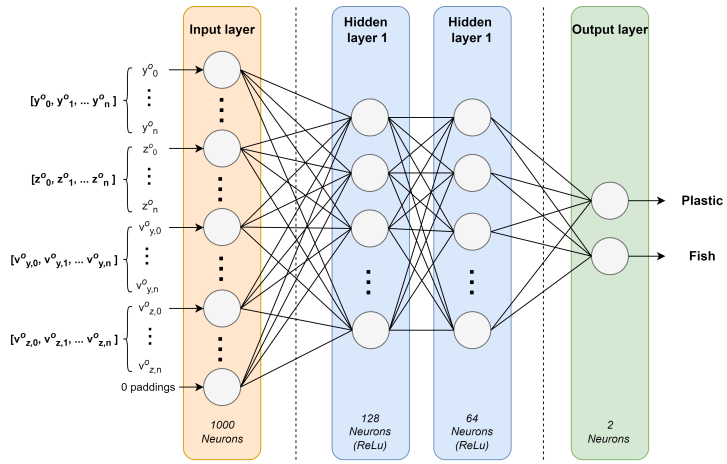


Figure 8.6: Architecture of the neural network used to classify the underwater moving objects

At the end of each training epoch, the performance of the neural network is

evaluated using the validation dataset. The highest accuracy is tracked throughout the training process. The final step is to apply the trained neural network that shows the highest accuracy for the validation set to the test dataset. A schematic overview of the entire process is illustrated in Figure 8.7.

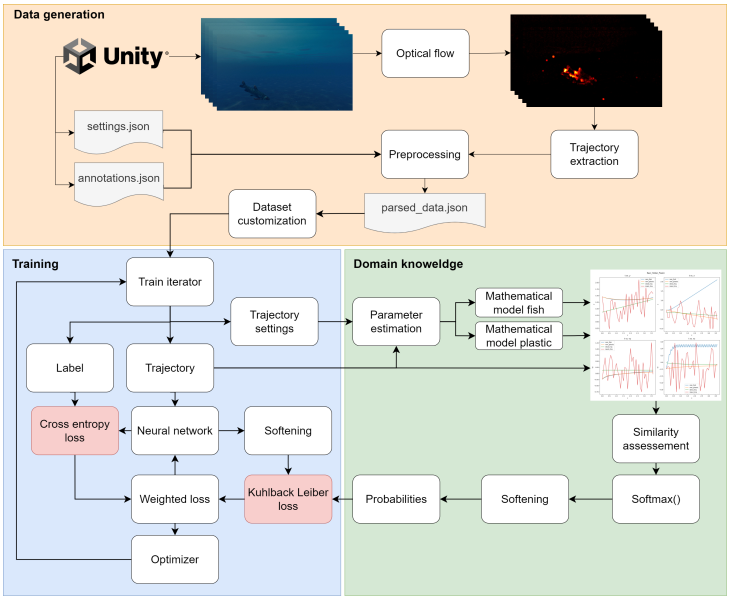


Figure 8.7: Overview of the proposed architecture for training the classifying neural network

8

8.6. Case Studies

This section presents various case studies that were conducted to evaluate the proposed architecture. Below a comprehensive overview of various scenarios is provided, distinguishing cases where the incorporation of domain knowledge proves beneficial and cases where the implementation of domain knowledge may not be justified. Two principal categories of scenarios are considered: those involving supervised learning and those involving semi-supervised learning. Three distinct cases (with regards to data types for training, validation, and testing) were considered per scenario:

1. Base data: Training, validation, and testing were all conducted on ideal data (i.e., data that is generated by the simulator without the addition of any noise). This case represents the conventional approach in machine learning where all the sets contain the same data distribution.

2. Low-quality data: Training and validation used optical flow data, whereas testing was performed on ideal data. This case has been designed to evaluate

the robustness of training on trajectories that are extracted using techniques such as optical flow that are widely used in above-water applications but that may face many challenges when applied in underwater scenarios.

3. Noisy data: Training and validation were conducted on ideal data that is augmented with Gaussian noise, whereas testing was performed on ideal data. This case simulates situations when sensor noise is present in the samples collected for the training process.

The three cases mentioned above (base data, low-quality data, and noisy data) allow to assess the impact of different data qualities on the performance of various object detection models, including a neural network with and without incorporated domain knowledge. The aim is to explore the potential advantages of incorporating domain-specific knowledge into the training procedure of a neural network, for various conditions. In addition to investigating the effects of the quality of the datasets, each case is also evaluated based on the dependability of the sample size of the training set. This evaluation is conducted by training two neural networks six times, with the training set size increasing from 100 to 600 samples in increments of 100. Both the validation and test datasets consistently hold 400 samples each.

To facilitate a robust assessment, a 10-fold cross-validation method is employed, with each fold containing 100 samples [225, 253]. Consequently, a total of 60 training runs per learning method are performed. A predefined seed is used to initialize the random neural network weights, ensuring that both the neural network with the domain knowledge module and the one without it start from the same initial conditions. Moreover, the neural networks are trained, validated, and tested on identical samples in order to maintain consistency across the experiments. The learning rate is maintained constant throughout the training process and is tuned per case of data type. The parameters of the mathematical models are estimated during the first epoch and are reused in subsequent epochs in order to improve the computational efficiency. This approach is viable because the mathematical models are static and remain unchanged during the training period.

The training of each neural network involves tuning the various hyperparameters of the model to optimize its performance. For this discussion, the batch size is kept equal to 1, and α in (8.6) is set equal to 0.5. The temperature parameter in (8.6) and the learning rate are manually fine-tuned.

Lastly, the robustness of the supervised and semi-supervised learning scenarios against low-quality annotations is examined. This is because annotations that are predominantly generated by humans are prone to errors. Thus, the resilience of both training processes is evaluated against incorrectly labeled data. Based on the existing literature [10, 251] and our experience, it is deemed that incorrectly labeling 20% of the samples in the training dataset is a realistic scenario.

8.6.1. SUPERVISED LEARNING

The initial setup that is investigated includes supervised learning, which encompasses the three previously discussed cases: base data, low-quality data, and noisy data. For each case, the accuracy of the neural networks, the impact of the size of the training

dataset on the performance of the trained model, and the consequence of using a poorly labeled dataset for training are examined.

The accuracy values obtained for the trained networks for the base data case (with training, validating, and testing processes done on ideal data) with a decreasing number of training samples are given in Figure 8.8a. It shows that increasing the number of data points for training the networks yields a higher performance for both of them. As the mathematical model is static, the number of data points does not impact its performance. The accuracy graphs for both networks are very similar, indicating that the domain knowledge is not impacting the learning significantly. Since a complex, non-linear mathematical model has been derived, which is assumed to have high accuracy, it shows the best performance among all. With a lower learning rate of 10^{-4}, the trend of the accuracy graph is further accentuated where both accuracy curves for the two neural network models almost fully coincide. This behavior can be explained by analyzing the probability distribution. The mathematical models closely approximate the ideal trajectory, positioning the probabilities at the extremes of the spectrum. Consequently, the information conveyed by these probabilities is identical to that contained in the labels, meaning the domain knowledge does not contribute additional insight in this case.

The accuracy values obtained for the trained models for the low-quality data case are presented in Figure 8.8c. In comparison to the results presented in Figure 8.8a, for the low-quality data case, the neural network with domain-knowledge-infused learning consistently outperforms the neural network without domain knowledge, with an average improvement of 2%. This is because the domain knowledge module instructs the neural network with domain knowledge on which trajectories to learn or disregard.

Finally, the accuracy values for the trained models in the noisy data case are shown Figure 8.8e, where the observed trend is consistent with the previous cases. In other words, the inclusion of domain knowledge consistently leads to an increased performance (around 2%) for the trained neural network. The domain knowledge module effectively prevents the neural network from overfitting to the Gaussian noise. In contrast, the absence of domain knowledge is marked by a (significant) performance decline at 600 data points, indicating overfitting.

The three cases explained above showed that the inclusion of domain knowledge will consistently improve the performance of the trained neural network, particularly for low-quality and noisy data. Similar accuracy results for lower numbers of samples for the training indicate the potential of the neural network with domain knowledge for modeling cases where limited data is available.

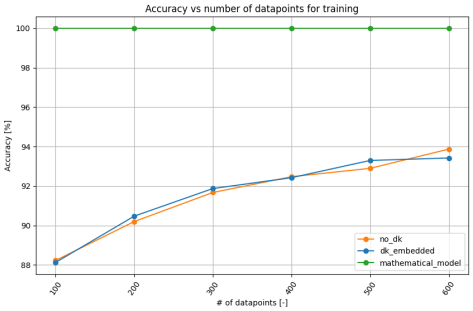
Next, the robustness of both neural networks in situations where 20% of the training dataset has been mislabelled is discussed. The accuracy values for the trained models with 20% mislabeled base data are presented in Figure 8.8b. In this scenario, the domain knowledge module demonstrates a distinct advantage. From 200 data points onwards, the neural network with domain knowledge consistently outperforms the neural network that does not incorporate domain knowledge. On average, a 2% improvement in performance is observed that diminishes with larger

training datasets. Via providing insights into the quality of the data, the domain knowledge module acts as a regularization, allowing the neural network to effectively identify and disregard incorrectly labeled trajectories. Consequently, incorporating domain knowledge enables the neural network to maintain accuracy even when trained on poorly annotated datasets, avoiding the accuracy loss that would occur if domain knowledge was not used.

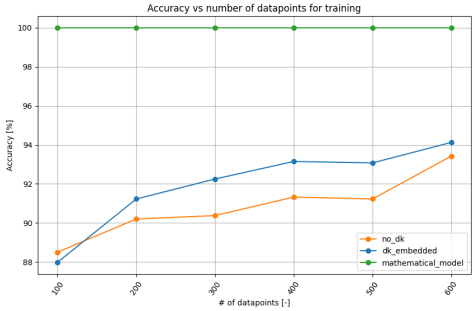
The impact of introducing erroneous labels to the low-quality data case is depicted in Figure 8.8d. It is evident that the incorporation of domain knowledge consistently increases the accuracy of the trained model across all training sample sizes. Although a slight reduction in the maximum accuracy is observed compared to the scenarios without erroneous labels, the performance of the neural network with domain knowledge remains approximately 2-5% higher than that of the neural network that is trained without domain knowledge included.

Finally, the accuracy values for the models trained with noisy data including 20% mislabelled data are shown in Figure 8.8f. A less consistent increase in the performance of the models is observed, with a notable decline at 400 data points. Moreover, while the neural network with domain knowledge shows a lower accuracy at 100 data points, it outperforms the neural network without domain knowledge at all other data sizes. In general, the performance across different training set sizes fluctuates significantly. These fluctuations are attributed to the high learning rate employed. The diminished performance at low data point counts is due to the domain knowledge module optimizing for two objectives: label accuracy and alignment with domain knowledge. This dual objective, especially when combined with a high learning rate, becomes particularly challenging with a limited amount of data.

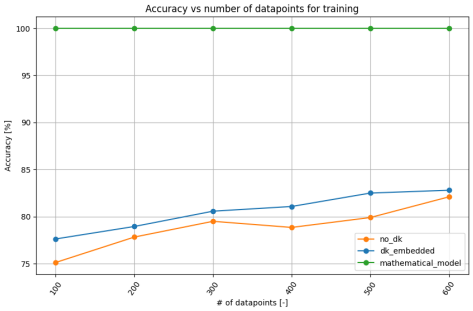
In summary, when utilizing supervised learning with different data distribution between the training and the test set, the integration of domain knowledge into the training procedure of a neural network enhances the accuracy of the trained architecture and improves its robustness against poorly annotated datasets.



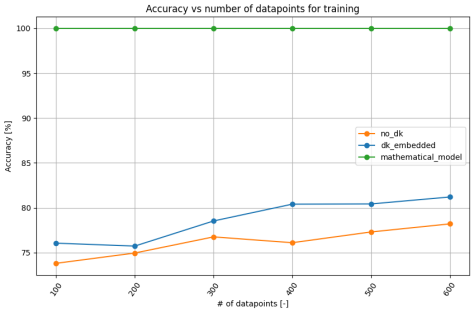
(a) Base data case: The accuracy when training, validating, and testing the two neural networks (with and without domain knowledge) on ideal data.



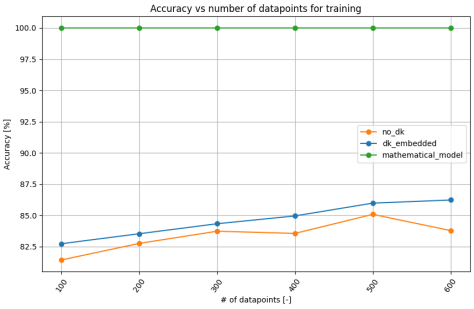
(b) Base data case: The accuracy when training, validating, and testing the two neural networks (with and without the domain knowledge) on ideal data with 20% of the samples mislabelled.



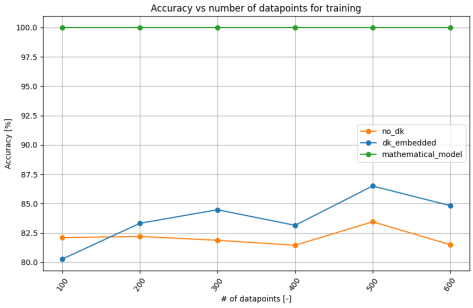
(c) Low-quality data case: The accuracy when training the two neural networks (with and without the domain knowledge) on optical flow data, validating them on optical flow data, and testing them on ideal data.



(d) Low-quality data case: The graph is constructed by training on optical flow data with 20% of the samples mislabelled, validating on optical flow data and testing on ideal data.



(e) Noisy data case: The accuracy when training the two neural networks (with and without the domain knowledge) on data including Gaussian noise, validating them on data including Gaussian noise, and testing on ideal data.



(f) Noisy data case: The accuracy values when training the two neural networks (with and without the domain knowledge) on data including Gaussian noise with 20% of the samples mislabelled, validating them on data including Gaussian noise, and testing them on ideal data.

Figure 8.8: Supervised learning: Comparison between the mathematical model and the neural networks trained with and without domain knowledge. The accuracy shown is based on the test set. Each row corresponds to a different type of data. The first column corresponds to training data with no errors in the labeling and the second column corresponds to training data with 20% mislabelled. The orange, blue, and green curves correspond to, respectively, the neural network without domain knowledge, the neural network with domain knowledge, and the domain knowledge module alone.

8.6.2. SEMI-SUPERVISED LEARNING

400 additional unlabeled data points of the same type as the labeled training dataset are incorporated into the training procedure. The analysis proceeds with the same cases as before, now augmented with the inclusion of unlabeled data.

The accuracy values for the trained networks when the base data case is used for training are shown in Figure 8.9a. It can be seen from this figure that the knowledge-infused training procedure significantly outperforms the training procedure without domain knowledge, particularly with lower numbers of data samples. The out-performance diminishes when larger numbers of data points are used for the training. In fact, as more data becomes available, the neural network trained without domain knowledge can approximate the underlying data distribution more accurately, ultimately approaching a similar level of performance as the network trained with domain knowledge. This emphasizes the advantage of employing semi-supervised learning together with domain knowledge, especially in scenarios where obtaining annotations comes with a large financial cost or is impossible in practice.

The accuracy values for the trained networks when low-quality data is used for training are shown in Figure 8.9c. In this case, a consistent out-performance by the neural network that is trained incorporating domain knowledge is observed. The neural network trained without the inclusion of domain knowledge does not ever catch up, while the neural network with domain knowledge outperforms it by 5-7%. This consistent advantage suggests that the low-quality (optical flow) data is significantly more complex to learn in the unsupervised way, compared to the base data case. It is expected that, with a sufficiently large number of data, the trend mirrors that of the base data case, albeit at a slower rate.

Finally, the accuracy values for the trained models when noisy data is used for training are illustrated in Figure 8.9e. This case yet shows a different trend compared to the above two cases. An out-performance of the neural network with domain knowledge is present, particularly for lower numbers of data points. Interestingly, at 400 data points —a threshold previously noted in the base data case as well— the training process that incorporates domain knowledge begins to learn the Gaussian noise characteristics. This is indicated by the decrease in the accuracy. Between 100 and 400 data points, the accuracy peaks for this process. It is concluded that with larger numbers of data points, some of the noise is learned by the neural network with domain knowledge. This effect could be minimized with a dynamic value for α that is adapted to balance the CE loss and KL divergence during the training process.

In summary, knowledge-infused training significantly boosts the accuracy of semi-supervised learning, especially with fewer training samples. This improvement is evident not only in the noisy data case but also in the base data case, demonstrating the efficacy of semi-supervised learning including domain knowledge across these various conditions.

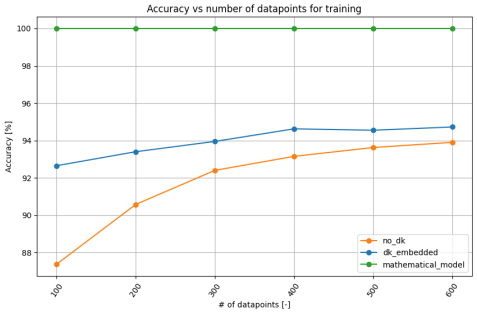
We also evaluate the robustness of the training with semi-supervised learning with and without domain knowledge when 20% mislabeled data is included in the training dataset. Figure 8.9b shows the accuracy of the neural networks trained with and without domain knowledge on the base data, where 20% of the data has been

wrongly labeled. A similar pattern for the accuracies as in the case of non-erroneous data is observed, with an enhanced performance per data point level and a plateau reached at 400 data points. The increased discrepancy, compared to the base data case, in the performance of the neural networks with knowledge-infused and standard training that is observed for larger numbers of data points, is attributed to the presence of the data with erroneous labels.

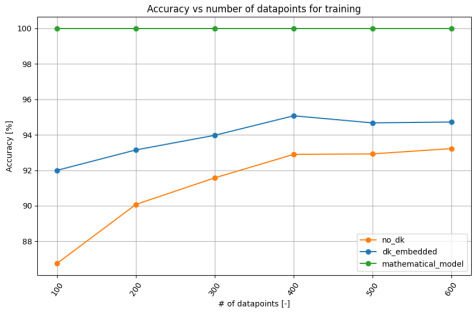
The obtained accuracies for the neural networks for the low-quality data, with 20% of the data being labeled wrongly, are shown in Figure 8.9d. A similar trend as for the low-quality data case without the erroneous labels is observed but with a smaller difference in the accuracies for both neural networks. This trend, which does not align with the conclusions drawn from the supervised learning, can be elucidated by examining the loss function, which is a linear combination of the CE loss and the KL divergence loss (see (8.6)). During the initial phase of training on the 400 unlabeled samples, the neural network trained with domain knowledge only minimizes the KL divergence loss. In the subsequent phase, the CE loss, which may include errors, is much larger than the KL divergence loss, and thus becomes more influential in the total loss. Consequently, the optimization process prioritizes the CE loss. This issue can be addressed by introducing α as an adaptable parameter, to ensure that the KL divergence loss maintains its effectiveness in the total loss. However, implementing a varying α is considered to be a topic for future research.

Finally, when 20% of the noisy data is mislabeled, the accuracy graphs are obtained as shown in Figure 8.9f. A similar trend as for the noisy data without wrong labeling is again observed, while at the 400 data points mark, the performance accuracy of the neural network that is trained without domain knowledge experiences a significant drop. This indicates that with a larger number of samples, erroneous elements are impacting further the learning of this neural network. This drop does not occur in the performance accuracy of the neural network that is trained including domain knowledge. It is noteworthy that the knowledge-guided learning process requires an average of 12 epochs to train, whereas the training process without domain knowledge requires 25 epochs. This reveals another advantage of combining semi-supervised learning with noisy data and domain knowledge for situations where the long training time should be avoided, such as in online learning.

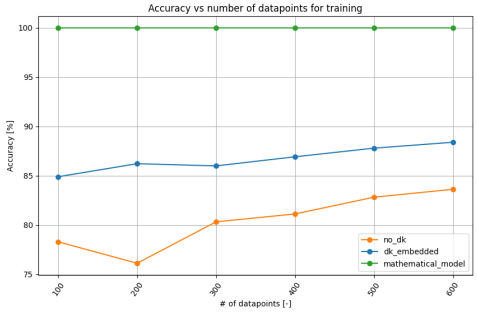
In summary, integrating domain knowledge allows for semi-supervised learning to significantly enhance the robustness of a neural network against low-quality and noisy data, especially including erroneous labels. The analysis across various cases and scenarios shows that domain knowledge-infused training consistently outperforms training that excludes domain knowledge, particularly in cases with lower numbers of data points and imperfect annotations. The analysis of the impact of the number of data points on the performance accuracy reveals that increasing the number of data points does not necessarily enhance the performance. Additionally, it was found that a subset of data points, combined with augmented training, can achieve similar performance accuracy as when larger datasets are used in training processes without domain knowledge included.



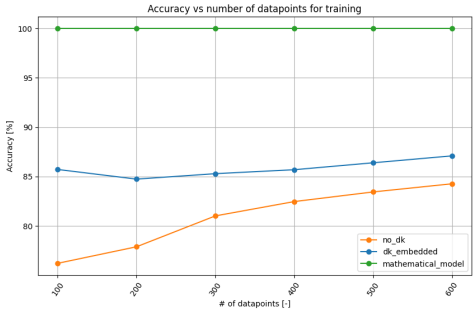
(a) Base data case: The accuracy when training, validating, and testing the two neural networks (with and without domain knowledge) on ideal data.



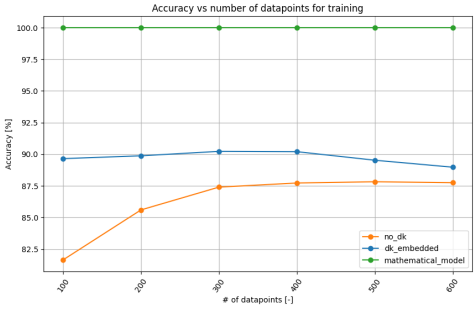
(b) Base data case: The accuracy when training, validating, and testing the two neural networks (with and without the domain knowledge) on ideal data with 20% of the samples mislabelled.



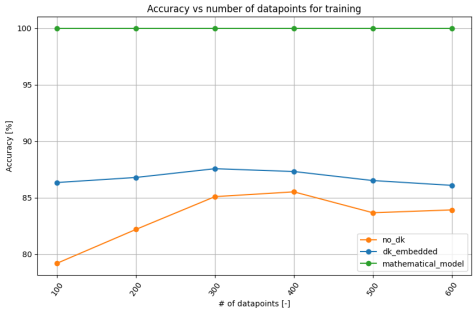
(c) Low-quality data case: The accuracy when training the two neural networks (with and without the domain knowledge) on optical flow data, validating them on optical flow data, and testing them on ideal data.



(d) Low-quality data case: The graph is constructed by training on optical flow data with 20% of the samples mislabelled, validating on optical flow data and testing on ideal data.



(e) Noisy data case: The accuracy when training the two neural networks (with and without the domain knowledge) on data including Gaussian noise, validating them on data including Gaussian noise, and testing on ideal data.



(f) Noisy data case: The accuracy values when training the two neural networks (with and without the domain knowledge) on data including Gaussian noise with 20% of the samples mislabelled, validating them on data including Gaussian noise, and testing them on ideal data.

Figure 8.9: Semi-supervised learning: Comparison between the mathematical model and the neural networks trained with and without domain knowledge. The accuracy shown is based on the test set. Each row corresponds to a different type of data. The first column corresponds to training data with no errors in the labeling and the second column corresponds to training data with 20% mislabelled. The orange, blue, and green curves correspond to, respectively, the neural network without domain knowledge, the neural network with domain knowledge, and the domain knowledge module alone.

8.6.3. COMPUTATIONAL EFFICIENCY

While the standalone domain knowledge module exhibits high accuracy, it is crucial to consider the intended final application, i.e., the real-time application of ROVs for autonomous underwater cleanup operations. The domain knowledge module relies on solving nonlinear differential equations, which is computationally intensive. Furthermore, as additional object classes are introduced, the module must incorporate new systems of differential equations, increasing computation time and making this approach less scalable. In contrast, this problem is not observed with the neural network. The domain knowledge module (solving the differential equations of each mathematical model) has a run time of 5.0381 seconds while the neural network has a run time of 0.0032 seconds, hence a difference of 15267 fold. The experiments are performed on NVIDIA GeForce RTX 3090. The neural network operates orders of magnitude faster than the domain knowledge module, making it the only feasible option for real-time on-board ROV applications.

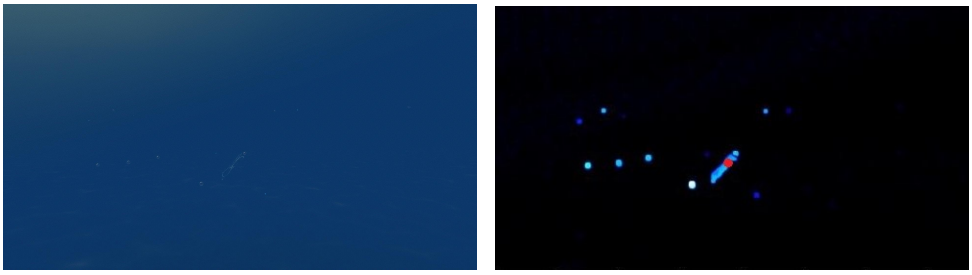
8.7. CONCLUSIONS AND FUTURE WORK

This chapter has introduced a novel pipeline that focuses on enhancing underwater object detection by incorporating knowledge about the dynamic behavior of the target objects into the training procedure of a neural network. To validate the effectiveness of this approach, a Unity-based simulator has been developed to generate high-quality datasets. A trajectory extraction module has also been designed using optical flow and a clustering algorithm as the first step of the proposed pipeline. Subsequently, two mathematical models have been formulated to describe the motion of the target objects. The probability distribution of the domain knowledge module has been computed using the mean squared error and the softmax function. This knowledge-infused training process has then been compared to standard methods without domain knowledge across various scenarios. The results demonstrate that training with domain knowledge notably enhances the robustness of the resulting neural network model against low-quality, noisy, or mislabeled data. Moreover, the inclusion of domain knowledge facilitates semi-supervised learning and consistently surpasses in performance accuracy the model that is trained solely based on data. In particular, for the case studies, an improvement of 10% in the accuracy on the test set has been observed in scenarios where only a limited number of annotations are available.

Future work should focus on enhancing the visual fidelity of the simulator, refining the trajectory extraction module i.e. with a Kalman filter and improving the training process by making the α parameter dynamic.

APPENDIX

8.A. MOTION EXTRACTION AND DATASET CREATION



(a) Image example containing a plastic beer holder.

(b) Generated mask.

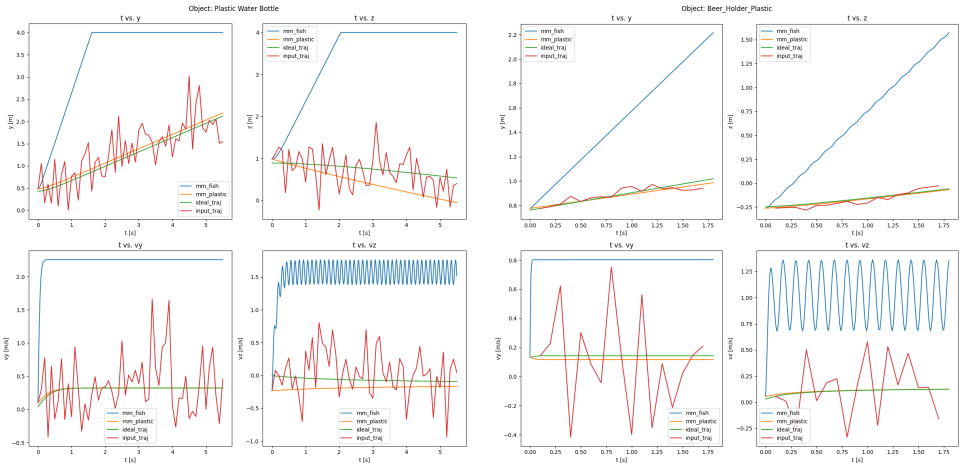
Figure 8.10: Original image and the optical flow output for a plastic beer holder.



(a) Image example containing fish.

(b) Generated mask.

Figure 8.11: Original image and the optical flow output for a fish.



(a) Gaussian noise trajectory sample

(b) Optical flow trajectory sample

Figure 8.12: Example trajectory samples.

9

Conclusions and Recommendations

9.1. Contributions

This PhD thesis presents advancements across multiple domains related to underwater debris detection and Deep Learning (DL) applications in challenging environments. Our main contributions can be summarized as follows:

- **Systematic categorization of the methods for DL for object detection and segmentation in videos**
  In the first part of this thesis, we have systematically organized and standardized DL approaches for underwater object detection and segmentation, focusing on object localization and pixel-level identification in video frames. Next, we have highlighted key challenges in pure DL techniques and proposed a novel hybrid framework that integrates DL techniques with domain-specific knowledge to enhance detection performance. The work has also reviewed existing methods that combine these approaches, providing insights into the potential benefits of hybrid architectures for future advancements in the field.

- **Creating and validating of the Seaclear marine debris dataset**
  We have introduced the Seaclear Marine Debris Dataset, a collection of annotated images with the aim to facilitate research in underwater object detection and segmentation. This is the first published dataset for underwater debris detection and instance segmentation in shallow waters. This dataset is crucial for training neural networks and making them capable of overcoming the domain shifts caused by variations in underwater conditions. We have also provided baseline results using state-of-the-art neural networks, emphasizing the challenges of generalization across different sites and conditions.

- **Demonstration of an autonomous multi-robot system for marine debris collection**
  We have presented the SeaClear system, a novel multi-robot platform designed for autonomous marine debris detection and collection. The system

integrates Artificial Intelligence (AI)-driven computer vision techniques with navigation and control algorithms to address the inherent challenges for robotics in underwater environments. We have also provided valuable insights gained from its real-world deployment, by presenting the practical challenges encountered and the lessons learned from applying the theoretical framework to demonstration scenarios.

- **Introducing novel evaluation aspects for underwater image enhancement within a complete computer vision pipeline**
  We have set the basis for evaluating underwater image enhancement methods as part of an integrated computer vision pipeline, rather than evaluating them as a standalone component. This approach addresses the current challenge, where Image Quality Assessment (IQA) metrics, especially the ones that do not take into account a reference image, have not yet been validated with respect to their relation to the performance of computer vision tasks.

- **Proposing a fusion scheme among data-based and model-based classifiers for robust classification on Out-of-Distribution (OOD) data**
  We have also addressed the challenge of poor generalization of deep neural networks on OOD data and proposed a model-based classifier that is designed based on the Moving Horizon Estimation (MHE) algorithm. The outputs of the two classifiers are fused using Dempster-Shafer (D-S) theory to achieve more reliable classification results, especially in the case of noisy and unseen data.

- **Improving DL-based object recognition through domain-specific knowledge:**
  We have proposed a novel approach for underwater object recognition by incorporating domain-specific knowledge into deep neural networks. Using knowledge distillation, a model-based classifier has been designed that serves as the teacher, while a simpler neural network is employed as the student. This method allows for both fully supervised and semi-supervised learning, and it results in enhanced accuracy and robustness, as it leverages the strengths of both approaches.

9.2. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORK

9.2.1. LIMITATIONS

We acknowledge some weaknesses of this work regarding the theoretical approaches that have been explored and some limitations that could emerge in the practical applications, which have not been fully addressed.

- **Reliance on Convolutional Neural Networks (CNNs)**
  This thesis solely focuses on CNN-based architectures for underwater object detection. While CNNs have demonstrated strong performance for real-time applications, newer DL algorithms, such as vision transformers [136], which have shown promising results in computer vision, have not been explored.

The decision to concentrate on CNNs was driven by their maturity and effectiveness in practical, real-time applications, justifying their use in this research. However, this choice comes with some limitations, as exploring more sophisticated DL architectures could provide improved performance.

- **Challenges with real-world deployment**
  Although field tests have been conducted to validate the implementation of several parts of this research work, the practical implementation of the perception module in diverse marine environments still contains many challenges. Factors such as hardware and sensor limitations and varying environmental conditions can significantly affect the performance. Due to these challenges, the new frameworks proposed in Chapter 7 and 8 serve as proof of concept in which simulation data rather than real-world data has been used.

- **Limited investigation of semi-supervised and unsupervised learning approaches**
  This thesis mainly focuses on supervised learning methods, which rely on labeled data. Semi-supervised or unsupervised learning techniques were beyond the scope of this work.

- **Premise of existing first-principles models**
  The focus of this thesis was not on developing new, complex mathematical models but rather on assuming the availability of such models and employing them for hybrid architectures coupling data-based with model-based techniques (Chapters 7 and 8). As a result, we did not explore the derivation of complex first-principles models, such as those that might describe the behavior of underwater animals, plants, or debris. In this way, we have concentrated on the concept of hybrid architectures but this decision could be viewed as a limitation in settings where more complex mathematical models are necessary.

- **Computational complexity and resource constraints**
  While methods like optical flow, as discussed in Chapter 8, are widely known for tracking applications, their computational complexity can limit real-time applications. This thesis does not extensively explore the trade-offs between computational efficiency and detection/tracking performance, which could hinder real-world implementation in some settings.

- **No exploration of 3D object detection and depth estimation**
  The scope of this thesis did not include the exploration of 3D object detection or depth estimation techniques. While these aspects could further enhance underwater perception and detection accuracy, they were considered outside the focus of this work.

9.2.2. RECOMMENDATIONS FOR FUTURE WORK

Despite the advancements made in this thesis, there remain several opportunities for future research. Some possible future research directions, based on the work

presented in this thesis are:

- **Exploration of semi-supervised unsupervised learning**
  Future research could investigate semi-supervised and unsupervised DL techniques, which depend less on large annotated datasets. By coupling these methods with logical rules, scientific knowledge, and causal relations, it would be possible to reduce the dependence on expensive, time-consuming data labeling and at the same time improve generalization, especially in dynamic and low-visibility underwater environments.

- **Specialized techniques for specific environments**
  The research in Chapter 6 presented the potential of specialized neural networks tailored to specific underwater conditions, such as low and high turbidity environments. A hierarchical classification framework could be developed to first classify the water quality, followed by successive classifications to detect litter, and finally identify the type of debris. This approach could improve the accuracy of detection in diverse environmental conditions.

- **Multi-sensor fusion**
  Incorporating data from multiple sensors, such as sonar, acoustic, or electromagnetic sensors, along with traditional vision-based methods, could improve detection accuracy, especially in environments with low visibility conditions like underwater. Future research could explore the integration of multi-sensor fusion techniques to create a more comprehensive system for underwater debris detection.

- **Employment of multiple mathematical models for real-time applications**
  The development of real-time systems for underwater debris detection could benefit from incorporating multiple mathematical models that account for different target objects. Future research should explore the trade-offs between computational complexity and model accuracy. By combining models that capture different behaviors (e.g., object motion, water currents), it may be possible to achieve more efficient and accurate detection.

- **Extension of simulation to real-world deployment**
  Chapters 7 and 8 presented architectures that combine deep-learning-based and model-based approaches, and they were validated on simulation data. To extend these methods to real-world applications, future research should focus on integrating real-world sensor data that includes synchronized visual and tracking data, and on developing accurate mathematical models that describe the behavior of underwater animals, plants, or debris. The design of the proposed architectures allows their adaptation to real-world sensor data.

- **Extension beyond underwater litter detection**
  While this PhD thesis has focused on underwater litter detection, the proposed methods could be extended and implemented to other underwater domains, such as underwater cable and equipment inspection, underwater archeology, and marine biology. Potential applications include monitoring the condition

of submerged structures, detecting and classifying underwater artifacts, or studying marine life.

- **Extension beyond underwater applications**
  The concepts explored in this PhD thesis could also be extended to domains outside underwater environments, addressing other areas requiring robust perception in challenging conditions. For example, the developed techniques could be further processed and employed in rescue robotics, where robot navigation and perception in adverse conditions are required, such as smoke-filled environments. Similarly, the classification and fusion frameworks developed in this PhD thesis could assist in medical imaging and in analyzing noisy scans, such as X-rays or ultrasounds, to enhance diagnostic accuracy. Furthermore, the methodologies presented in this thesis could contribute to space exploration, enabling robotic systems to perform tasks such as planet surface exploration, where conditions are highly uncertain.

BIBLIOGRAPHY

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi. "A review of uncertainty quantification in deep learning: techniques, applications, and challenges". In: *Inf. Fusion* 76.C (Dec. 2021), pp. 243–297. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.05.008.

[2] A. Alessandri, M. Baglietto, and G. Battistelli. "Moving-horizon state estimation for nonlinear discrete-time systems: new stability results and approximation schemes". In: *Automatica* 44.7 (2008), pp. 1753–1765. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2007.11.020.

[3] K. O. Amer, M. Elbouz, A. Alfalou, C. Brosseau, and J. Hajjami. "Enhancing underwater optical imaging by using a low-pass polarization filter". In: *Opt. Express* 27.2 (Jan. 2019), pp. 621–643. DOI: 10.1364/OE.27.000621.

[4] C. O. Ancuti, C. Ancuti, C. De Vleeschouwer, and P. Bekaert. "Color balance and fusion for underwater image enhancement". In: *IEEE Transactions on Image Processing* 27.1 (2018), pp. 379–393. DOI: 10.1109/TIP.2017.2759252.

[5] C. Ancuti, C. O. Ancuti, T. Haber, and P. Bekaert. "Enhancing underwater images and videos by fusion". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 81–88. DOI: 10.1109/CVPR.2012.6247661.

[6] C. Anitescu, B. İsmail Ateş, and T. Rabczuk. *Physics-informed neural networks: theory and applications*. Cham: Springer International Publishing, 2023, pp. 179–218. DOI: 10.1007/978-3-031-36644-4_5.

[7] "Application of uncertainty quantification to artificial intelligence in healthcare: A review of last decade (2013–2023)". In: *Computers in Biology and Medicine* 165 (2023), p. 107441. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2023.107441.

[8] Y. Ba, R. Chen, Y. Wang, L. Yan, B. Shi, and A. Kadambi. "Physics-based neural networks for shape from polarization". In: *CoRR* abs/1903.10210 (2019). arXiv: 1903.10210. URL: http://arxiv.org/abs/1903.10210.

[9] A. Babić, F. Ferreira, N. Kapetanović, N. Mišković, M. Bibuli, G. Bruzzone, C. Motta, R. Ferretti, A. Odetti, M. Caccia, S. Aracri, and F. De Pascalis. "Cooperative marine litter detection and environmental monitoring using heterogeneous robotic agents". In: *OCEANS 2023 - Limerick*. 2023, pp. 1–6. DOI: 10.1109/OCEANSLimerick52467.2023.10244657.

[10] S. Bai, S. Zhou, Z. Qin, L. Wang, and N. Zheng. "Robust noisy label learning via two-stream sample distillation". In: *arXiv preprint arXiv:2404.10499* (2024). URL: https://arxiv.org/abs/2404.10499.

[11] C. Barrelet, M. Chaumont, G. Subsol, V. Creuze, and M. Gouttefarde. "From TrashCan to UNO: deriving an underwater image dataset to get a more consistent and balanced version". In: *Pattern Recognition, Computer Vision, and Image Processing. ICPR 2022 International Workshops and Challenges: Montreal, QC, Canada, August 21–25, 2022, Proceedings, Part III*. Montréal, QC, Canada: Springer-Verlag, 2023, pp. 403–414. ISBN: 978-3-031-37730-3. DOI: 10.1007/978-3-031-37731-0_30.

[12] B. Basu, S. Sannigrahi, A. Sarkar Basu, and F. Pilla. "Development of novel classification algorithms for detection of floating plastic debris in coastal waterbodies using multispectral sentinel-2 remote sensing imagery". In: *Remote Sensing* 13.8 (Apr. 2021), p. 1598. ISSN: 2072-4292. DOI: 10.3390/rs13081598.

[13] H. Bay, T. Tuytelaars, and L. Van Gool. "SURF: speeded up robust features". In: *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[14] G. Bertasius and L. Torresani. "Classifying, segmenting, and tracking object instances in video with mask propagation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. DOI: 10.1109/CVPR42600.2020.00976.

[15] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. "Fully-convolutional siamese networks for object tracking". In: *Computer Vision – ECCV Workshops*. 2016, pp. 850–865. ISBN: 978-3-319-48881-3.

[16] C. Bhalerao. *YOLO v8! The real state-of-the-art?* Medium. Jan. 2023. URL: https://medium.com/mlearning-ai/yolo-v8-the-real-state-of-the-art-eda6c86a1b90.

[17] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, and H. Ghayvat. "CNN variants for computer vision: history, architecture, application, challenges and future scope". In: *Electronics* 10.20 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10202470. URL: https://www.mdpi.com/2079-9292/10/20/2470.

[18] B. Bingham, C. Agüero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar. "Toward maritime robotic simulation in Gazebo". In: *OCEANS 2019 MTS/IEEE Seattle*. 2019, pp. 1–10. DOI: 10.23919/OCEANS40490.2019.8962724.

[19] C. M. Bishop. *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag, 2006. ISBN: 0387310738.

[20] D. Bolya, C. Zhou, F. Xiao, and Y. J. Leee. "YOLACT++: Better real-time instance segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020). DOI: 10.1109/TPAMI.2020.3014297.

[21] H. Booth, W. Ma, and O. Karakus. "High-precision density mapping of marine debris and floating plastics via satellite imagery". In: (2022). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2210.05468.

[22] S. B. Borrelle, J. Ringma, K. L. Law, C. C. Monnahan, L. Lebreton, A. McGivern, E. Murphy, J. Jambeck, G. H. Leonard, M. A. Hilleary, M. Eriksen, H. P. Possingham, H. De Frond, L. R. Gerber, B. Polidoro, A. Tahir, M. Bernard, N. Mallos, M. Barnes, and C. M. Rochman. "Predicted growth in plastic waste exceeds efforts to mitigate plastic pollution". In: *Science* 369.6510 (Sept. 2020), pp. 1515–1518. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aba3656.

[23] H. Bourlard and Y. Kamp. "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological cybernetics* 59.4-5 (1988), pp. 291–294. ISSN: 0340-1200. DOI: 10.1007/bf00332918. URL: https://doi.org/10.1007/bf00332918.

[24] G. Bradski. "The OpenCV library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[25] A. Broad, M. Jones, and T. Y. Lee. "Recurrent multi-frame single shot detector for video object detection". In: *BMVC*. 2018.

[26] J. Bu, M. Elhamod, C. S. M. Redell, W. Lee, and A. Karpatne. "Learning neural networks with competing physics objectives: an application in quantum mechanics". In: *CoRR* abs/2007.01420 (2020). arXiv: 2007.01420. URL: https://arxiv.org/abs/2007.01420.

[27] D. Bulic, D. Tolić, and I. Palunko. "Beam-based tether dynamics and simulations using finite element model". In: *IFAC-PapersOnLine* 55.15 (2022). 6th IFAC Conference on Intelligent Control and Automation Sciences ICONS 2022, pp. 154–159. DOI: https://doi.org/10.1016/j.ifacol.2022.07.624.

[28] S. Caelles, A. Pumarola, F. Moreno-Noguer, A. Sanfeliu, and L. Van Gool. "Fast video object segmentation with spatio-temporal GANs". In: *CoRR* abs/1903.12161 (2019). arXiv: 1903.12161. URL: http://arxiv.org/abs/1903.12161.

[29] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF: binary robust independent elementary features". In: *Computer Vision – ECCV 2010*. Springer Berlin Heidelberg, 2010, pp. 778–792.

[30] H. Chen, J. Chen, and J. Ding. "Data evaluation and enhancement for quality improvement of machine learning". In: *IEEE Transactions on Reliability* 70.2 (2021), pp. 831–847. DOI: 10.1109/TR.2021.3070863.

[31] X. Chen, Y. Lu, Z. Wu, J. Yu, and L. Wen. "Reveal of domain effect: how visual restoration contributes to object detection in aquatic scenes". In: *CoRR* abs/2003.01913 (2020). arXiv: 2003.01913. URL: https://arxiv.org/abs/2003.01913.

[32] X. Chen, J. Yu, S. Kong, Z. Wu, X. Fang, and L. Wen. "Towards real-time advancement of underwater visual quality with GAN". In: *IEEE Transactions on Industrial Electronics* 66.12 (Dec. 2019), pp. 9350–9359. DOI: 10.1109/tie.2019.2893840.

[33] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *Proceedings Conference on Empirical Methods in Natural Language Processing, EMNLP*. 2014. DOI: 10.3115/v1/d14-1179.

[34] C. Clarke, M. Hall, G. Mittal, Y. Yu, S. Sajeev, J. Mars, and M. Chen. "Rule by example: harnessing logical rules for explainable hate speech detection". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 364–376. DOI: 10.18653/v1/2023.acl-long.22.

[35] D. Comaniciu, V. Ramesh, and P. Meer. "Real-time tracking of non-rigid objects using mean shift". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2000, pp. 142–149. DOI: 10.1109/CVPR.2000.854761.

[36] M. Córdova, A. Pinto, C. C. Hellevik, S. A.-A. Alaliyat, I. A. Hameed, H. Pedrini, and R. d. S. Torres. "Litter detection with deep learning: a comparative study". In: *Sensors* 22.2 (Jan. 2022), p. 548. ISSN: 1424-8220. DOI: 10.3390/s22020548.

[37] M. Córdova, A. Pinto, C. C. Hellevik, S. A.-A. Alaliyat, I. A. Hameed, H. Pedrini, and R. d. S. Torres. *PlastOPol: a dataset for litter detection*. 2022. DOI: https://doi.org/10.5281/zenodo.5829155.

[38] C. Cortes and V. Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[39] A. Cózar, F. Echevarría, J. I. González-Gordillo, X. Irigoien, B. Úbeda, S. Hernández-León, A. T. Palma, S. Navarro, J. García-de-Lomasx, A. Ruiz, F.-P. M. L., and C. M. Duarte. "Plastic debris in the open ocean". In: *Proceedings of the National Academy of Sciences* 111.28 (2014), pp. 10239–10244. DOI: 10.1073/pnas.1314705111.

[40] A. Coźar, M. Sanz-Martín, E. Martí, J. I. Gonzaĺez-Gordillo, B. Ubeda, J. Gaĺvez, X. Irigoien, and C. M. Duarte. "Plastic accumulation in the mediterranean sea". In: *PLOS ONE* 10.4 (Apr. 2015). Ed. by E. V. Thuesen, e0121762. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0121762.

[41] M. D'heer, A. Ilioudi, B. De Schutter, A. Dabiri, and A. Jamshidnejad. "Domain-informed neural networks for detecting underwater moving objects with noisy datasets". In: *Knowledge-Based Systems* (2025). to be submitted.

[42] J. Dai, Y. Li, K. He, and J. Sun. "R-FCN: Object detection via region-based fully convolutional networks". In: *CoRR* abs/1605.06409 (2016). arXiv: 1605.06409. URL: http://arxiv.org/abs/1605.06409.

[43] R. A. Dakhil and A. R. H. Khayeat. "Review on deep learning techniques for underwater object detection". In: *Data Science and Machine Learning*. Academy and Industry Research Collaboration Center (AIRCC), Sept. 2022. DOI: 10.5121/csit.2022.121505.

[44] T. Dash, S. Chitlangia, A. Ahuja, and A. Srinivasan. “A review of some techniques for inclusion of domain-knowledge into deep neural networks”. In: *Scientific Reports* 12 (2022), p. 1040. DOI: 10.1038/s41598-021-04590-0.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei. “ImageNet: a large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL: IEEE, June 2009, pp. 248–255. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848.

[46] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei. “Relation distillation networks for video object detection”. In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 7022–7031. DOI: 10.1109/ICCV.2019.00712.

[47] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei. “Single shot video object detector”. In: *IEEE Transactions on Multimedia* 23 (2021), pp. 846–858. DOI: 10.1109/TMM.2020.2990070.

[48] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun. “RepVGG: making VGG-style convnets great again”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13733–13742.

[49] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagtk, D. Cremers, and T. Brox. “FlowNet: Learning optical flow with convolutional networks”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316.

[50] D. Duarte, U. Andriolo, and G. Gonçalves. “Addressing the class imbalance problem in the automatic image classification of coastal litter from orthophotos derived from UAS imagery”. en. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-3-2020 (Aug. 2020), pp. 439–445. ISSN: 2194-9050. DOI: 10.5194/isprs-annals-V-3-2020-439-2020.

[51] A. Đuraš, A. Ilioudi, B. Wolf, I. Palunko, and B. De Schutter. *SeaClear marine debris detection and segmentation dataset*. en. 2024. DOI: https://doi.org/10.4121/4F1DFF25-E157-4399-A5D4-478055461689.V1.

[52] A. Đuraš, A. Ilioudi, B. Wolf, I. Palunko, and B. De Schutter. *Seaclear marine debris detection and segmentation dataset*. 2024. DOI: https://doi.org/10.4121/4F1DFF25-E157-4399-A5D4-478055461689.V1.

[53] A. Đuraš, M. Sukno, and I. Palunko. “Recovering the 3D UUV position using UAV imagery in shallow-water environments”. In: *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2022, pp. 948–954. DOI: 10.1109/ICUAS54217.2022.9836195.

[54] A. Đuraš, B. J. Wolf, A. Ilioudi, I. Palunko, and B. De Schutter. “A dataset for detection and segmentation of underwater marine debris in shallow waters”. In: *Scientific Data* 11.1 (Aug. 2024), p. 921. ISSN: 2052-4463. DOI: 10.1038/s41597-024-03759-2.

[55] T. Emmerik and A. Schwarz. “Plastic debris in rivers”. en. In: *WIREs Water* 7.1 (Jan. 2020). ISSN: 2049-1948, 2049-1948. DOI: 10.1002/wat2.1398.

[56] M. Eriksen, L. C. M. Lebreton, H. S. Carson, M. Thiel, C. J. Moore, J. C. Borerro, F. Galgani, P. G. Ryan, and J. Reisser. “Plastic pollution in the world’s oceans: more than 5 trillion plastic pieces weighing over 250,000 tons afloat at sea”. In: *PLoS ONE* 9.12 (Dec. 2014). Ed. by H. G. Dam, e111913. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0111913.

[57] G. Farnebäck. “Two-frame motion estimation based on polynomial expansion”. In: *Image Analysis*. Ed. by J. Bigun and T. Gustavsson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370. ISBN: 978-3-540-45103-7. DOI: https://doi.org/10.1007/3-540-45103-X_50.

[58] M. Fava. *Ocean plastic pollution: an overview: data and statistics*. Accessed October 31, 2024. May 2022. URL: https://oceanliteracy.unesco.org/plastic-pollution-ocean/#:~:text=Currently%5C%2C%5C%20there%5C%20are%5C%20about%5C%2050,ends%5C%20u%5C%20forming%5C%20garbage%5C%20patches.

[59] C. Feichtenhofer, A. Pinz, and A. Zisserman. “Detect to track and track to detect”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3057–3065. DOI: 10.1109/ICCV.2017.330.

[60] P. Felzenszwalb, D. McAllester, and D. Ramanan. “A discriminatively trained, multiscale, deformable part model”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587597.

[61] F. Fernández de la Mata, A. Gijón, M. Molina-Solana, and J. Gómez-Romero. “Physics-informed neural networks for data-driven simulation: advantages, limitations, and opportunities”. In: *Physica A: Statistical Mechanics and its Applications* 610 (2023), p. 128415. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2022.128415.

[62] T. I. Fossen. *Guidance and control of ocean vehicles*. New York: Wiley, 1995.

[63] Froggreen. *Fish*. Accessed October 31, 2024. 2017. URL: https://sketchfab.com/3d-models/fish-ae9089d355d244aebd9abee4da7d35af.

[64] C. Fu, R. Liu, X. Fan, P. Chen, H. Fu, W. Yuan, M. Zhu, and Z. Luo. “Rethinking general underwater object detection: Datasets, challenges, and solutions”. In: *Neurocomputing* 517 (2023), pp. 243–256. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.10.039.

[65] M. Fulton, J. Hong, M. J. Islam, and J. Sattar. “Robotic detection of marine litter using deep visual detection models”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5752–5758.

[66] M. S. Fulton, J. Hong, and J. Sattar. *Trash-ICRA19: A bounding box labeled dataset of underwater trash*. 2020. DOI: https://doi.org/10.13020/x0qn-y082.

[67] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. *Robot operating system (ROS): the complete reference (volume 1)*. Cham: Springer International Publishing, 2016. Chap. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. ISBN: 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23.

[68] Y. Gal and Z. Ghahramani. "Dropout as a bayesian approximation: representing model uncertainty in deep learning." In: *Proceedings of the 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1050–1059.

[69] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. "Informed RRT*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 2997–3004. DOI: 10.1109/IROS.2014.6942976.

[70] B. Gao and M. W. Spratling. "More robust object tracking via shape and motion cue integration". In: *Signal Processing* (2022), p. 108628.

[71] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera. "Big data preprocessing: methods and prospects". In: *Big Data Analytics* 1 (2016), pp. 1–22.

[72] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[73] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets robotics: the KITTI dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[74] T. Georgiou, Y. Liu, W. Chen, and M. S. Lew. "A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision". In: *International Journal of Multimedia Information Retrieval* 9 (2019), pp. 135–170.

[75] J. J. Gibson. *The ecological approach to visual perception*. Houghton Mifflin, 1979.

[76] R. Girdhar, J. J. Carreira, C. Doersch, and A. Zisserman. "Video action transformer network". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 244–253. DOI: 10.1109/CVPR.2019.00033.

[77] R. Girshick. "Fast R-CNN". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[78] D. González-Fernández, A. Cózar, G. Hanke, J. Viejo, C. Morales-Caselles, R. Bakiu, D. Barceló, F. Bessa, A. Bruge, M. Cabrera, J. Castro-Jiménez, M. Constant, R. Crosti, Y. Galletti, A. E. Kideys, N. Machitadze, J. Pereira de Brito, M. Pogojeva, N. Ratola, J. Rigueira, E. Rojo-Nieto, O. Savenko, R. I. Schöneich-Argent, G. Siedlewicz, G. Suaria, and M. Tourgeli. "Floating macrolitter leaked from Europe into the ocean". en. In: *Nature Sustainability* 4.6 (June 2021), pp. 474–483. ISSN: 2398-9629. DOI: 10.1038/s41893-021-00722-6.

[79] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN: 9780262035613. URL: https://books.google.nl/books?id=Np9SDQAAQBAJ.

[80] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2672–2680.

[81] C. Goutte and E. Gaussier. "A probabilistic interpretation of precision, recall, and f-score, with implication for evaluation". In: *Advances in Information Retrieval*. Ed. by D. E. Losada and J. M. Fernández-Luna. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 345–359. ISBN: 978-3-540-31865-1.

[82] M. Gouttefarde, M. Rodriguez, C. Barrelet, P.-E. Hervé, V. Creuze, J. Gorrotxategi, A. Oyarzabal, D. Culla, D. Sallé, O. Tempier, N. Ferrari, M. Chaumont, and G. Subsol. "The robotic seabed cleaning platform: an underwater cable-driven parallel robot for marine litter removal". In: *Cable-Driven Parallel Robots*. Ed. by S. Caro, A. Pott, and T. Bruckmann. Cham: Springer, 2023, pp. 430–441. ISBN: 978-3-031-32322-5.

[83] K. Grauman and T. Darrell. "The pyramid match kernel: discriminative classification with sets of image features". In: *Tenth IEEE International Conference on Computer Vision (ICCV)*. 2005, pp. 1458–1465. DOI: 10.1109/ICCV.2005.239.

[84] H. S. Greenwald and C. K. Oertel. "Future directions in machine learning". In: *Frontiers in Robotics and AI* 3 (2017), p. 79. ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00079.

[85] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. "LSTM: a search space odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. DOI: 10.1109/TNNLS.2016.2582924.

[86] R. L. Gregory. *Eye and brain: the psychology of seeing*. Fifth. Princeton and Oxford: Princeton University Press, 1997, p. 275.

[87] Y. Gu, L. Wang, Z. Wang, Y. Liu, M.-M. Cheng, and S.-P. Lu. "Pyramid constrained self-attention network for fast video salient object detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[88] V. Gudivada, A. Apon, and J. Ding. "Data quality considerations for big data and machine learning: going beyond data cleaning and transformations". In: *International Journal on Advances in Software* 10 (2017), pp. 1–20.

[89] C. Guo, B. Fan, J. Gu, Q. Zhang, S. Xiang, V. Prinet, and C. Pan. "Progressive sparse local attention for video object detection". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019).

[90] D. Hall, F. Dayoub, J. Skinner, H. Zhang, D. Miller, P. Corke, G. Carneiro, A. Angelova, and N. Sünderhauf. "Probabilistic object detection: Definition and evaluation". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 1020–1029. DOI: 10.1109/WACV45572.2020.9093599.

[91] F. Han, J. Yao, H. Zhu, and C. Wang. "Marine organism detection and classification from underwater vision based on the deep CNN method". In: *Mathematical Problems in Engineering* 2020 (Feb. 2020), pp. 1–11. ISSN: 1024-123X, 1563-5147. DOI: 10.1155/2020/3937580.

[92] J. Han and C. Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning". In: *From Natural to Artificial Neural Computation*. Springer Berlin Heidelberg, 1995, pp. 195–201. ISBN: 978-3-540-49288-7.

[93] B. Hariharan, P. Arbelaez, R. B. Girshick, and J. Malik. "Simultaneous detection and segmentation". In: *CoRR* abs/1407.1808 (2014). arXiv: 1407.1808. URL: http://arxiv.org/abs/1407.1808.

[94] C. Harris and M. Stephens. "A combined corner and edge detector". In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151.

[95] E. Haseltine and J. Rawlings. "Critical evaluation of extended Kalman filtering and moving-horizon estimation". In: *Industrial and Engineering Chemistry Research* 44 (Apr. 2005), pp. 2451–2460. DOI: 10.1021/ie0343081.

[96] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2020), pp. 386–397. DOI: 10.1109/TPAMI.2018.2844175.

[97] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

[98] P. Henderson and V. Ferrari. "End-to-end training of object class detectors for mean average precision". In: *Computer Vision – ACCV 2016*. Ed. by S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato. Cham: Springer International Publishing, 2017, pp. 198–213.

[99] B. E. Henriques, M. Baglioni, and A. Jamshidnejad. "Camera-based mapping in search-and-rescue via flying and ground robot teams". In: *Machine Vision and Applications* 35.5 (2024), p. 117. DOI: 10.1007/s00138-024-01594-4.

[100] S. Heshmati-Alamdari, A. Eqtami, G. C. Karras, D. V. Dimarogonas, and K. J. Kyriakopoulos. "A self-triggered visual servoing model predictive control scheme for under-actuated underwater robotic vehicles". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 3826–3831.

[101] S. Heshmati-Alamdari, G. C. Karras, P. Marantos, and K. J. Kyriakopoulos. "A robust model predictive control approach for autonomous underwater vehicles operating in a constrained workspace". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6183–6188. DOI: 10.1109/ICRA.2018.8460918.

[102] C. Hetang, H. Qin, S. Liu, and J. Yan. "Impression network for video object detection". In: *CoRR* abs/1712.05896 (2017). arXiv: 1712.05896. URL: http://arxiv.org/abs/1712.05896.

[103] G. E. Hinton and T. J. Sejnowski. *Learning and relearning in Boltzmann machines*. Cambridge, MA, USA: MIT Press, 1986, pp. 282–317. ISBN: 026268053X.

[104] G. E. Hinton, S. Osindero, and Y.-W. Teh. "A fast learning algorithm for deep belief nets". In: *Neural Computation* 18.7 (July 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. eprint: https://direct.mit.edu/neco/article-pdf/18/7/1527/816558/neco.2006.18.7.1527.pdf.

[105] G. E. Hinton and R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313 (2006), pp. 504–507.

[106] G. E. Hinton, O. Vinyals, and J. Dean. "Distilling the knowledge in a neural network". In: *CoRR* abs/1503.02531 (2015). URL: http://arxiv.org/abs/1503.02531.

[107] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[108] G. Hollows and N. James. *Understanding focal length and field of view*. Accessed October 31, 2024. 2023. URL: https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-focal-length-and-field-of-view/.

[109] J. Hong, M. Fulton, and J. Sattar. *Trashcan: a semantically-segmented dataset towards visual detection of marine debris*. 2020. DOI: 10.48550/arXiv.2007.08097. arXiv: 2007.08097 [cs.CV].

[110] J. Hong, M. S. Fulton, and J. Sattar. *TrashCan 1.0: An instance-segmentation labeled dataset of trash observations*. 2020. DOI: https://doi.org/10.13020/g1gx-y834.

[111] B. K. Horn and B. G. Schunck. "Determining optical flow". In: *Artificial Intelligence* 17.1 (1981), pp. 185–203. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(81)90024-2.

[112] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. "OctoMap: an efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous Robots* 34.3 (2013), pp. 189–206. DOI: 10.1007/s10514-012-9321-0.

[113] B. J. Hou and Z. H. Zhou. “Learning with interpretable structure from gated RNN”. In: *IEEE Transactions on Neural Networks and Learning Systems* 31 (2020), pp. 2267–2279.

[114] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. “Harnessing deep neural networks with logic rules”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 2410–2420. DOI: 10.18653/v1/P16-1228.

[115] A. Ilioudi, A. Dabiri, B. J. Wolf, and B. De Schutter. “Deep learning for object detection and segmentation in videos: Toward an integration with domain knowledge”. In: *IEEE Access* 10 (2022), pp. 34562–34576. DOI: 10.1109/ACCESS.2022.3162827.

[116] A. Ilioudi, L. Gharavi, K. ter Burg, A. Dabiri, and B. De Schutter. “Deep learning and physics-based models fusion for classification in out-of-distribution scenarios”. In: *Information Sciences* (2025). to be submitted.

[117] A. Ilioudi, S. Sosnowski, E. Banken, P. Bevanda, J. Brüdigam, L. Buşoniu, Y. Chardard, C. Delea, B. De Schutter, A. Đuraš, C. Hertel-ten Eikelder, S. Heshmati-Alamdari, V.-M. Maer, I. Palunko, I. Pozniak, V. Prkačin, and D. Tolić. “The SeaClear system: an intelligent multi-robot solution for autonomous cleanup of marine debris on the seabed”. In: *Engineering Applications of Artificial Intelligence* (May 2024). Submitted, under review.

[118] A. Ilioudi, B. J. Wolf, A. Dabiri, and B. De Schutter. “Towards establishing an automated selection framework for underwater image enhancement methods”. In: *OCEANS 2023 - Limerick*. 2023, pp. 1–6. DOI: 10.1109/OCEANSLimerick52467.2023.10244710.

[119] A. Isobe and S. Iwasaki. “The fate of missing ocean plastics: are they just a marine environmental problem?” In: *Science of The Total Environment* 825 (2022), p. 153935. ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2022.153935.

[120] M. Isogawa, Y. Yuan, M. O’Toole, and K. Kitani. “Optical non-line-of-sight physics-based 3D human pose estimation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 7011–7020. DOI: 10.1109/CVPR42600.2020.00704.

[121] A. Jabbar, X. Li, and B. Omar. “A survey on generative adversarial networks: variants, applications, and training”. In: *ACM Comput. Surv.* 54.8 (Oct. 2021). ISSN: 0360-0300. DOI: 10.1145/3463475.

[122] G. Jakovljevic, M. Govedarica, and F. Alvarez-Taboada. “A deep learning model for automatic plastic mapping using unmanned aerial vehicle (UAV) data”. In: *Remote Sensing* 12.9 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12091515.

[123] J. R. Jambeck, R. Geyer, C. Wilcox, T. R. Siegler, M. Perryman, A. Andrady, R. Narayan, and K. L. Law. “Plastic waste inputs from land into the ocean”. In: *Science* 347.6223 (Feb. 2015), pp. 768–771. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1260352.

[124] JAMSTEC. *Deep-sea debris database. Japan Agency for Marine Earth Science and Technology*. 2018. URL: https://www.godac.jamstec.go.jp/dsdebris/e/index.html.

[125] A. Jesus, C. Zito, C. Tortorici, E. Roura, and G. De Masi. "Underwater object classification and detection: first results and open challenges". In: *OCEANS 2022 - Chennai*. IEEE, Feb. 2022. DOI: 10.1109/oceanschennai45887.2022.9775417.

[126] T. Jia, Z. Kapelan, R. de Vries, P. Vriend, E. C. Peereboom, I. Okkerman, and R. Taormina. "Deep learning for detecting macroplastic litter in water bodies: A review". In: *Water Research* 231 (Mar. 2023), p. 119632. ISSN: 00431354. DOI: 10.1016/j.watres.2023.119632.

[127] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. S. Steinbach, and V. Kumar. *Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles*. May 2019, pp. 558–566. ISBN: 978-1-61197-567-3. DOI: 10.1137/1.9781611975673.63.

[128] X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. S. Steinbach, and V. Kumar. "Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles". In: *CoRR* abs/2001.11086 (2020). arXiv: 2001.11086. URL: https://arxiv.org/abs/2001.11086.

[129] L. Jiang, Y. Wang, Q. Jia, S. Xu, Y. Liu, X. Fan, H. Li, R. Liu, X. Xue, and R. Wang. "Underwater species detection using channel sharpening attention". en. In: *Proceedings of the 29th ACM International Conference on Multimedia*. Virtual Event China: ACM, Oct. 2021, pp. 4259–4267. ISBN: 978-1-4503-8651-7. DOI: 10.1145/3474085.3475563.

[130] G. Jocher, A. Chaurasia, and J. Qiu. *YOLO by Ultralytics*. 2023. URL: https://github.com/ultralytics/ultralytics.

[131] A. Kahana, E. Turkel, S. Dekel, and D. Givoli. "Obstacle segmentation based on the wave equation and deep learning". In: *Journal of Computational Physics* 413 (2020), p. 109458. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2020.109458.

[132] R. E. Kalman. "A new approach to linear filtering and prediction problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[133] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N.Samatova, and V. Kumar. "Theory-guided data science: A new paradigm for scientific discovery from data". In: *IEEE Transactions on Knowledge and Data Engineering* 29.10 (Oct. 2017), pp. 2318–2331. ISSN: 1041-4347. DOI: 10.1109/tkde.2017.2720168.

[134] A. Karpatne, W. Watkins, J. S. Read, and V. Kumar. *Physics-guided neural networks (PGNN): an application in lake temperature modeling*. Taylor & Francis, 2022. Chap. 15, pp. 353–372. DOI: 10.1201/9781003143376-15.

[135] A. Kendall and Y. Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 5574–5584.

[136] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. “Transformers in vision: a survey”. In: *ACM Computing Surveys* 54.10s (Sept. 2022). DOI: 10.1145/3505244.

[137] K. Kikaki, I. Kakogeorgiou, P. Mikeli, D. E. Raitsos, and K. Karantzalos. “MARIDA: a benchmark for marine debris detection from Sentinel-2 remote sensing data”. In: *PLOS ONE* 17.1 (Jan. 2022). Ed. by B. K. Veettil, e0262247. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0262247.

[138] Y. Kim, C. Denton, L. Hoang, and A. M. Rush. “Structured attention networks”. In: *CoRR* abs/1702.00887 (2017). arXiv: 1702.00887. URL: http://arxiv.org/abs/1702.00887.

[139] D. P. Kingma and J. Ba. “Adam: a method for stochastic optimization”. In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[140] D. P. Kingma and M. Welling. “Auto-encoding variational Bayes”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014. arXiv: http://arxiv.org/abs/1312.6114v10 [stat.ML].

[141] D. Klink, A. Peytavin, and L. Lebreton. “Size dependent transport of floating plastics modeled in the global ocean”. In: *Frontiers in Marine Science* 9 (July 2022), p. 903134. ISSN: 2296-7745. DOI: 10.3389/fmars.2022.903134.

[142] J. F. Kolen and S. C. Kremer. *Gradient flow in recurrent nets: The difficulty of learning long-term dependencies*. 2001, pp. 237–243. DOI: 10.1109/9780470544037.ch14.

[143] S. Kong, M. Tian, C. Qiu, Z. Wu, and J. Yu. “IWSCR: an intelligent water surface cleaner robot for collecting floating garbage”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.10 (2021), pp. 6358–6368. DOI: 10.1109/TSMC.2019.2961687.

[144] J. Kossen, K. Stelzner, M. Hussing, C. Voelcker, and K. Kersting. “Structured object-aware physics prediction for video modeling and planning”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=B1e-kxSKDH.

[145] A. Kumar and S. Srivastava. “Object detection system based on convolution neural networks using single shot multi-box detector”. In: *Procedia Computer Science* 171 (2020), pp. 2610–2617. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.04.283.

[146] B. M. Lake and M. Baroni. “Generalization without systematicity: on the compositional skills of sequence-to-sequence recurrent networks”. In: *ICML*. 2018.

[147] S. Lazebnik, C. Schmid, and J. Ponce. “Beyond bags of features: spatial pyramid matching for recognizing natural scene categories”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2006, pp. 2169–2178. DOI: 10.1109/CVPR.2006.68.

[148] L. Lebreton, B. Slat, F. Ferrari, B. Sainte-Rose, J. Aitken, R. Marthouse, S. Hajbane, S. Cunsolo, A. Schwarz, A. Levivier, K. Noble, P. Debeljak, H. Maral, R. Schoeneich-Argent, R. Brambini, and J. Reisser. “Evidence that the Great Pacific garbage patch is rapidly accumulating plastic”. In: *Scientific Reports* 8.1 (Mar. 2018), p. 4666. ISSN: 2045-2322. DOI: 10.1038/s41598-018-22939-w.

[149] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[150] C. Li, S. Tang, J. Yan, and T. Zhou. “Low-light image enhancement based on quasi-symmetric correction functions by fusion”. In: *Symmetry* 12.9 (2020). ISSN: 2073-8994. DOI: 10.3390/sym12091561.

[151] C. Y. Li and A. Cavallaro. “Cast-GAN: learning to remove colour cast from underwater images”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 1083–1087. DOI: 10.1109/ICIP40778.2020.9191157.

[152] C. Y. Li and A. Cavallaro. “On the limits of perceptual quality measures for enhanced underwater images”. In: *2022 IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 4148–4152. DOI: 10.1109/ICIP46576.2022.9897840.

[153] C. Y. Li, R. Mazzon, and A. Cavallaro. *Underwater image filtering: methods, datasets and evaluation*. Dec. 2020. arXiv: 2012.12258 [cs.CV].

[154] C. Li, C. Guo, W. Ren, R. Cong, J. Hou, S. Kwong, and D. Tao. “An underwater image enhancement benchmark dataset and beyond”. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 4376–4389. DOI: 10.1109/TIP.2019.2955241.

[155] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei. *YOLOv6: A single-stage object detection framework for industrial applications*. arXiv:2209.02976 [cs]. Sept. 2022.

[156] G. Li, H. Liu, U. Müller, C. Voesenek, and J. Van Leeuwen. “Fishes regulate tail-beat kinematics to minimize speed-specific cost of transport”. English. In: *Proceedings of the Royal Society B: Biological Sciences* 288.1964 (Dec. 2021). ISSN: 0962-8452. DOI: 10.1098/rspb.2021.1601.

[157] K. Li, W. Ma, U. Sajid, Y. Wu, and G. Wang. “Object detection with convolutional neural networks”. In: *CoRR* abs/1912.01844 (2019). arXiv: 1912.01844. URL: http://arxiv.org/abs/1912.01844.

[158] Q. Li, C. Zhang, Q. Hu, P. Zhu, H. Fu, and L. Chen. “Stabilizing multispectral pedestrian detection with evidential hybrid fusion”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 34.4 (2024), pp. 3017–3029. DOI: 10.1109/TCSVT.2023.3306870.

[159] X. Li, Y. Dai, Y. Ge, J. Liu, Y. Shan, and L. DUAN. “Uncertainty modeling for out-of-distribution generalization”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=6HN7LHyzGgC.

[160] C. Lieshout, K. Oeveren, T. Emmerik, and E. Postma. “Automated river plastic monitoring using deep learning and cameras”. In: *Earth and Space Science* 7.8 (Aug. 2020). ISSN: 2333-5084, 2333-5084. DOI: 10.1029/2019EA000960.

[161] C. C. Lin, Y. Hung, R. Feris, and L. He. “Video instance segmentation tracking with a modified VAE architecture”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 13144–13154. DOI: 10.1109/CVPR42600.2020.01316.

[162] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. “Microsoft COCO: common objects in context”. In: *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48.

[163] T. Lindeberg. “Feature detection with automatic scale selection”. In: *International Journal of Computer Vision* 30.2 (1998), pp. 79–116.

[164] Z. C. Lipton and J. Berkowitz. “A critical review of recurrent neural networks for sequence learning”. In: *CoRR* abs/1506.00019 (2015). arXiv: 1506.00019. URL: http://arxiv.org/abs/1506.00019.

[165] C. Liu, H. Li, S. Wang, M. Zhu, D. Wang, X. Fan, and Z. Wang. “A dataset and benchmark of underwater object detection for robot picking”. In: *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. Shenzhen, China: IEEE, July 2021, pp. 1–6. ISBN: 978-1-66544-989-2. DOI: 10.1109/ICMEW53276.2021.9455997.

[166] C. Liu, Z. Wang, S. Wang, T. Tang, Y. Tao, C. Yang, H. Li, X. Liu, and X. Fan. “A new dataset, Poisson GAN and AquaNet for underwater object grabbing”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.5 (May 2022), pp. 2831–2844. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2021.3100059.

[167] D. Liu, Y. Cui, Y. Chen, J. Zhang, and B. Fan. “Video object detection for autonomous driving: motion-aid feature calibration”. In: *Neurocomputing* 409 (2020), pp. 1–11. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.05.027.

[168] K. Liu, Q. Sun, D. Sun, M. Yang, and N. Wang. “Underwater target detection based on improved YOLOv7”. In: (2023). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2302.06939.

[169] M. Liu, M. Zhu, M. White, Y. Li, and D. Kalenichenko. “Looking fast and slow: Memory-guided mobile video object detection”. In: *CoRR* abs/1903.10172 (2019). arXiv: 1903.10172. URL: http://arxiv.org/abs/1903.10172.

[170] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. "SSD: single shot multibox detector". In: *Computer Vision – ECCV 2016*. Ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Vol. 9905. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46447-3. DOI: 10.1007/978-3-319-46448-0_2.

[171] X. Liu and A. Goldsmith. "alman filtering with partial observation losses". In: *2004 43rd IEEE Conference on Decision and Control (CDC)*. Vol. 4. 2004, pp. 4180–4186.

[172] Z. Liu X.and Deng and Y. Yang. "Recent progress in semantic image segmentation". In: 52.2 (Aug. 2019), pp. 1089–1106. ISSN: 0269-2821. DOI: 10.1007/s10462-018-9641-3.

[173] L. Ljung. *System identification: theory for the user*. 2nd. Upper Saddle River, New Jersey: Prentice-Hall, 1999.

[174] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. 1999, pp. 1150–1157.

[175] D. G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. DOI: 10.1023/b:visi.0000029664.99615.94.

[176] Y. Lu, C. Lu, and C.-K. Tang. "Online video object detection using association LSTM". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2363–2371. DOI: 10.1109/ICCV.2017.257.

[177] V.-M. Maer, L. Tamás, and L. Buşoniu. "Underwater robot pose estimation using acoustic methods and intermittent position measurements at the surface". In: *2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. 2022, pp. 1–6.

[178] S. Mallick. *Geometry of image formation*. LearnOpenCV, OpenCV. Accessed October 31, 2024. 2020. URL: https://learnopencv.com/geometry-of-image-formation/.

[179] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach. "UUV simulator: a Gazebo-based package for underwater intervention and multi-robot simulation". In: *OCEANS 2016 MTS/IEEE Monterey*. 2016, pp. 1–8. DOI: 10.1109/OCEANS.2016.7761080.

[180] G. Marcus. "Deep learning: a critical appraisal". In: *CoRR* abs/1801.00631 (2018). URL: https://arxiv.org/pdf/1801.00631.pdf.

[181] G. F. Marcus. "The next decade in AI: four steps towards robust artificial intelligence". In: *ArXiv* abs/2002.06177 (2020).

[182] *Marine debris: understanding, preventing, and mitigating the significant adverse impacts on marine and coastal biodiversity*. Secretariat of the Convention on Biological Diversity, Oct. 2016. ISBN: 9789292256258.

[183] C. Martin, Q. Zhang, D. Zhai, X. Zhang, and C. M. Duarte. “Enabling a large-scale assessment of litter along Saudi Arabian red sea shores by combining drones and machine learning”. en. In: *Environmental Pollution* 277 (May 2021), p. 116730. ISSN: 02697491. DOI: 10.1016/j.envpol.2021.116730.

[184] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei. “Deep learning for visual tracking: a comprehensive survey”. In: *CoRR* abs/1912.00535 (2019). arXiv: 1912.00535. URL: http://arxiv.org/abs/1912.00535.

[185] R. McAllister, G. Kahn, J. Clune, and S. Levine. “Robustness to out-of-distribution inputs via task-aware generative uncertainty”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 2083–2089.

[186] J. McGlade, I. Samy Fahim, D. Green, P. Landrigan, A. Andrady, M. Costa, R. Geyer, R. Gomes, A. Tan Shau Hwai, J. Jambeck, D. Li, C. Rochman, P. Ryan, M. Thiel, R. Thompson, K. Townsend, and A. Turra. *From pollution to solution: a global assessment of marine litter and plastic pollution*. Tech. rep. United Nations Environment Programme, 2021.

[187] T. Moore and D. Stouch. “A generalized extended Kalman filter implementation for the robot operating system”. In: *Advances in Intelligent Systems and Computing*. Vol. 302. 2016, pp. 335–348. DOI: 10.1007/978-3-319-08338-4_25.

[188] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem. “TrackingNet: s large-scale dataset and benchmark for object tracking in the wild”. In: *CoRR* abs/1803.10794 (2018). arXiv: 1803.10794. URL: http://arxiv.org/abs/1803.10794.

[189] W. K. Mutlag, S. K. Ali, Z. M. Aydam, and B. H. Taher. “Feature extraction methods: a Review”. In: *Journal of Physics: Conference Series* 1591.1 (July 2020), p. 012028. DOI: 10.1088/1742-6596/1591/1/012028.

[190] M. A. Nabian and H. Meidani. “Physics-informed regularization of deep neural networks”. In: *CoRR* abs/1810.05547 (2018). arXiv: 1810.05547. URL: http://arxiv.org/abs/1810.05547.

[191] National Oceanic and Atmospheric Administration Marine Debris Program. *Report on marine debris impacts on coastal and benthic habitats*. 2016.

[192] Natural seabed. https://naturalseabed.com/en/. Accessed: 2024-03-25. 2024.

[193] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh. “Deep learning vs. traditional computer vision”. In: *Advances in Computer Vision*. Springer International Publishing, 2020, pp. 128–144. ISBN: 978-3-030-17795-9.

[194] S. W. Oh, J.-Y. Lee, K. Sunkavalli, and S. J. Kim. “Fast video object segmentation by reference-guided mask propagation”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[195] OpenCV. *K-means clustering*. Accessed October 31, 2024. 2023. URL: https://docs.opencv.org/4.x/d1/d5c/tutorial_py_kmeans_opencv.html.

[196] A. Oussidi and A. Elhassouny. "Deep generative models: survey". In: *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*. 2018, pp. 1–8. DOI: 10.1109/ISACV.2018.8354080.

[197] D. Pagac, E. Nebot, and H. Durrant-Whyte. "An evidential approach to map-building for autonomous vehicles". In: *IEEE Transactions on Robotics and Automation* 14.4 (1998), pp. 623–629. DOI: 10.1109/70.704234.

[198] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[199] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191.

[200] K. Panetta, C. Gao, and S. Agaian. "Human-visual-system-inspired underwater image quality measures". In: *IEEE Journal of Oceanic Engineering* 41.3 (July 2016), pp. 541–551. DOI: 10.1109/joe.2015.2469915.

[201] K. Panetta, A. Samani, and S. Agaian. "Choosing the optimal spatial domain measure of enhancement for mammogram images". In: *International Journal of Biomedical Imaging* 2014 (2014), pp. 1–8. DOI: 10.1155/2014/937849.

[202] R. Paolucci, F. Gatti, M. Infantino, C. Smerzini, A. Özcebe, and M. Stupazzini. "Broadband ground motions from 3D physics-based numerical simulations using artificial neural networks". In: *Bulletin of the Seismological Society of America* 108 (2018), pp. 1272–1286. ISSN: 0037-1106. DOI: 10.1785/0120170293. eprint: https://pubs.geoscienceworld.org/bssa/article-pdf/108/3A/1272/4189975/bssa-2017293.1.pdf.

[203] A. Papakonstantinou, M. Batsaris, S. Spondylidis, and K. Topouzelis. "A citizen science unmanned aerial system data acquisition protocol and deep learning techniques for the automatic detection and mapping of marine litter concentrations in the coastal zone". In: *Drones* 5.1 (Jan. 2021), p. 6. ISSN: 2504-446X. DOI: 10.3390/drones5010006.

[204] C. Patel, D. Bhatt, U. Sharma, R. Patel, S. Pandya, K. Modi, N. Cholli, A. Patel, U. Bhatt, M. A. Khan, S. Majumdar, M. Zuhair, K. Patel, S. A. Shah, and H. Ghayvat. "DBGC: dimension-based generic convolution block for object recognition". In: *Sensors* 22.5 (2022). ISSN: 1424-8220. DOI: 10.3390/s22051780.

[205] C. I. Patel, S. Garg, T. Zaveri, and A. Banerjee. "Top-down and bottom-up cues based moving object detection for varied background video sequences". In: *Adv. MultiMedia* (2014). ISSN: 1687-5680. DOI: 10.1155/2014/879070.

[206] J. Pearl and D. Mackenzie. *The book of why: the new science of cause and effect*. 1st. USA: Basic Books, Inc., 2018. ISBN: 046509760X.

[207] M. Pedersen, J. Bruslund Haurum, R. Gade, and T. B. Moeslund. “Detection of marine animals in a new underwater dataset with varying visibility”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.

[208] Y.-T. Peng, Y.-C. Lin, W.-Y. Peng, and C.-Y. Liu. “Blurriness-guided underwater salient object detection and data augmentation”. In: *IEEE Journal of Oceanic Engineering* (2024), pp. 1–15. DOI: 10.1109/JOE.2023.3344154.

[209] D. V. Politikos, A. Adamopoulou, G. Petasis, and F. Galgani. “Using artificial intelligence to support marine macrolitter research: a content analysis and an online database”. In: *Ocean & Coastal Management* 233 (Feb. 2023), p. 106466. ISSN: 09645691. DOI: 10.1016/j.ocecoaman.2022.106466.

[210] D. V. Politikos, E. Fakiris, A. Davvetas, I. A. Klampanos, and G. Papatheodorou. “Automatic detection of seafloor marine litter using towed camera images and deep learning”. en. In: *Marine Pollution Bulletin* 164 (Mar. 2021), p. 111974. ISSN: 0025326X. DOI: 10.1016/j.marpolbul.2021.111974.

[211] D. K. Prasad, C. K. Prasath, D. Rajan, L. Rachmawati, E. Rajabaly, and C. Quek. “Challenges in video based object detection in maritime scenario using computer vision”. In: *arXiv preprint arXiv:1608.01079* (2016).

[212] M. Prats, J. Pérez, J. J. Fernández, and P. J. Sanz. “An open source tool for simulation and supervision of underwater intervention missions”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 2577–2582. DOI: 10.1109/IROS.2012.6385788.

[213] P. F. Proença and P. Simões. *TACO: Trash annotations in context for litter detection*. 2020. arXiv: 2003.06975 [cs.CV].

[214] P. F. Proença and P. Simões. *TACO: trash annotations in context for litter detection*. arXiv:2003.06975 [cs]. Mar. 2020. DOI: https://doi.org/10.48550/arXiv.2003.06975.

[215] U. N. E. Programme. *World leaders set sights on plastic pollution*. Accessed: 2024-11-01. 2020. URL: https://www.unep.org/news-and-stories/story/world-leaders-set-sights-plastic-pollution.

[216] R. Rai and C. K. Sahu. “Driven by data or derived through physics? A review of hybrid physics guided machine learning techniques with cyber-physical system (CPS) focus”. In: *IEEE Access* 8 (2020), pp. 71050–71073. DOI: 10.1109/ACCESS.2020.2987324.

[217] M. Raissi, P. Perdikaris, and G. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045.

[218] T. Ramalho and M. Miranda. "Density estimation in representation space to predict model uncertainty". In: *Engineering Dependable and Secure Machine Learning Systems*. Ed. by O. Shehory, E. Farchi, and G. Barash. Cham: Springer International Publishing, 2020, pp. 84–96.

[219] RanMarine technology. https://www.ranmarine.io/. Accessed: 2024-03-25. 2024.

[220] C. Rao, J. Rawlings, and D. Mayne. "Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations". In: *IEEE Transactions on Automatic Control* 48.2 (2003), pp. 246–258. DOI: 10.1109/TAC.2002.808470.

[221] S. Rath. *YOLOv6 custom dataset training – underwater trash detection*. English. LearnOpenCV. Nov. 2022. URL: https://learnopencv.com/yolov6-custom-dataset-training/.

[222] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: unified, real-time object detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 779–788. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.91.

[223] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2015.

[224] R. Rodineliussen. "Caring for water: underwater waste, trash diving, and publicity in stockholm". In: *kritisk etnografi: Swedish Journal of Anthropology* 4.2 (2021), pp. 73–92. ISSN: 2003-1173. DOI: https://doi.org/10.33063/diva-463875.

[225] J. D. Rodriguez, A. Perez, and J. A. Lozano. "Sensitivity analysis of k-fold cross validation in prediction error estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.3 (2010), pp. 569–575. DOI: 10.1109/TPAMI.2009.187.

[226] M. Rosynski and L. Buşoniu. "A simulator and first reinforcement learning results for underwater mapping". In: *Sensors* 22.14 (2022), p. 5384.

[227] RoutineStudio. *Plastic water bottle*. Accessed October 31, 2024. 2020. URL: https://sketchfab.com/3d-models/plastic-water-bottle-731efe2635c9472c9c1e4fdb1f8fbd13.

[228] D. Rozumnyi, J. Matas, F. Šroubek, M. Pollefeys, and M. R. Oswald. "FMODetect: robust detection of fast moving objects". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3541–3549.

[229] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F. F. Li. "ImageNet large scale visual recognition challenge". In: *Int. J. Comput. Vision* 115.3 (Dec. 2015), pp. 211–252. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0816-y.

[230] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. “LabelMe: a database and web-based tool for image annotation”. In: *International Journal of Computer Vision* 77.1–3 (Oct. 2007), pp. 157–173. ISSN: 1573-1405. DOI: 10.1007/s11263-007-0090-8. URL: http://dx.doi.org/10.1007/s11263-007-0090-8.

[231] M. Sadoughi and C. Hu. “Physics-based convolutional neural network for fault diagnosis of rolling element bearings”. In: *IEEE Sensors Journal* 19.11 (2019), pp. 4181–4192. DOI: 10.1109/JSEN.2019.2898634.

[232] R. Salakhutdinov and G. Hinton. “Deep Boltzmann machines”. In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, Apr. 2009, pp. 448–455. URL: https://proceedings.mlr.press/v5/salakhutdinov09a.html.

[233] R. Salakhutdinov and H. Larochelle. “Efficient learning of deep Boltzmann machines”. In: *AISTATS*. 2010.

[234] A. Sánchez-Ferrer, J. J. Valero-Mas, A. J. Gallego, and J. Calvo-Zaragoza. “The CleanSea set: a benchmark corpus for underwater debris detection and recognition”. In: *Iberian Conference on Pattern Recognition and Image Analysis*. 2022.

[235] A. Sánchez-Ferrer, J. J. Valero-Mas, A. J. Gallego, and J. Calvo-Zaragoza. “An experimental study on marine debris location and recognition using object detection”. In: *Pattern Recognition Letters* 168 (Apr. 2023), pp. 154–161. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2022.12.019.

[236] T. Sasaki, S. Azuma, S. Matsuda, A. Nagayama, M. Ogido, H. Saito, and Y. Hanafusa. “JAMSTEC e-library of deep-sea images (J-EDI) realizes a virtual journey to the earth’s unexplored deep ocean”. In: *AGU Fall Meeting Abstracts*. Vol. 2016. Dec. 2016, IN53C-1911, IN53C–1911.

[237] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. “Grad-CAM: visual explanations from deep networks via gradient-based localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[238] S. Seo, S. Ö. Arik, J. Yoon, X. Zhang, K. Sohn, and T. Pfister. “Controlling neural networks with rule representations”. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS ’21. Red Hook, NY, USA: Curran Associates Inc., 2024. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/5cd5058bca53951ffa7801bcdf421651-Paper.pdf.

[239] I. Ševo and A. Avramović. “Convolutional neural network based automatic object detection on aerial images”. In: *IEEE Geoscience and Remote Sensing Letters* 13.5 (2016), pp. 740–744. DOI: 10.1109/LGRS.2016.2542358.

[240] Z. Shen, J. Liu, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui. “Towards out-of-distribution generalization: a survey”. In: *ArXiv* abs/2108.13624 (2021).

[241] M. Shvets, W. Liu, and A. Berg. "Leveraging long-range temporal relationships between proposals for video object detection". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9755–9763. DOI: 10.1109/ICCV.2019.00985.

[242] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *CoRR* abs/1409.1556 (2015).

[243] S. Singh, A. Prasad, K. Srivastava, and S. Bhattacharya. "Object motion detection methods for real-time video surveillance: A survey with empirical evaluation". In: *Smart Systems and IoT: Innovations in Computing*. 2020, pp. 663–679. ISBN: 978-981-13-8406-6.

[244] S. K. Singh, R. Yang, A. Behjat, R. Rai, S. Chowdhury, and I. Matei. "PI-LSTM: Physics-infused long short-term memory network". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 2019, pp. 34–41. DOI: 10.1109/ICMLA.2019.00015.

[245] A. Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". English. In: *IEE Proceedings F (Radar and Signal Processing)* 140 (2 Apr. 1993), 107–113(6). ISSN: 0956-375X. URL: https://digital-library.theiet.org/content/journals/10.1049/ip-f-2.1993.0015.

[246] I. Sobel and G. Feldman. "A 3×3 isotropic gradient operator for image processing". In: *Pattern Classification and Scene Analysis* (Jan. 1973), pp. 271–272.

[247] Á. Solé Gómez, L. Scandolo, and E. Eisemann. "A learning approach for river debris detection". en. In: *International Journal of Applied Earth Observation and Geoinformation* 107 (Mar. 2022), p. 102682. ISSN: 15698432. DOI: 10.1016/j.jag.2022.102682.

[248] K. Song, J.-Y. Jung, S. H. Lee, and S. Park. "A comparative study of deep learning-based network model and conventional method to assess beach debris standing-stock". en. In: *Marine Pollution Bulletin* 168 (July 2021), p. 112466. ISSN: 0025326X. DOI: 10.1016/j.marpolbul.2021.112466.

[249] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. Lau, and M.-H. Yang. "VITAL: visual tracking via adversarial learning". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[250] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[251] M. Staats, M. Thamm, and B. Rosenow. "Reevaluating loss functions: enhancing robustness to label noise in deep learning models". In: *ArXiv* abs/2306.05497 (2023). URL: https://api.semanticscholar.org/CorpusID:259129809.

[252] R. Stewart and S. Ermon. “Label-free supervision of neural networks with physics and domain knowledge”. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI’17. San Francisco, California, USA: AAAI Press, 2017, pp. 2576–2582.

[253] M. Stone. “Cross-validatory choice and assessment of statistical predictions”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2 (1974), pp. 111–133. DOI: 10.1111/j.2517-6161.1974.tb00994.x.

[254] H. Suk. *An introduction to neural networks and deep learning*. English. Elsevier Inc., Jan. 2017, pp. 3–24. ISBN: 9780128104088. DOI: 10.1016/B978-0-12-810408-8.00002-X.

[255] M. Sukno and I. Palunko. “Hand-crafted features for floating plastic detection”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 3378–3383. DOI: 10.1109/IROS47612.2022.9981320.

[256] F. Sultana, A. Sufian, and P. Dutta. “A review of object detection models based on convolutional neural network”. In: *Intelligent Computing: Image Processing Based Applications* (2020), pp. 1–16. ISSN: 2194-5365. DOI: 10.1007/978-981-15-4288-6_1.

[257] M. Sultana, A. Mahmood, S. Javed, and S. K. Jung. “Unsupervised RGBD video object segmentation using GANs”. In: *CoRR* abs/1811.01526 (2018). arXiv: 1811.01526. URL: http://arxiv.org/abs/1811.01526.

[258] L. Sun, H. Gao, S. Pan, and J. X. Wang. “Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data”. In: *Computer Methods in Applied Mechanics and Engineering* (2020). ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2019.112732.

[259] X. Sun, P. Wu, and S. C. Hoi. “Face detection using deep learning: An improved faster RCNN approach”. In: *Neurocomputing* 299 (2018), pp. 42–50. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.03.030.

[260] P. Szleg, P. Barczyk, B. Maruszczak, S. Zielinski, and E. Szymańska. “Simulation environment for underwater vehicles testing and training in Unity3D”. In: *Annual Meeting of the IEEE Industry Applications Society*. 2022.

[261] R. Tao, E. Gavves, and A. W. M. Smeulders. “Siamese instance search for tracking”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1420–1429. DOI: 10.1109/CVPR.2016.158.

[262] G. Tata. *Deepplastic: an open source image dataset for epipelagic marine plastic detection*. Apr. 2021. DOI: 10.5281/ZENODO.5562940.

[263] T.-H. Teng, A.-H. Tan, and J. M. Zurada. “Self-organizing neural networks integrating domain knowledge and reinforcement learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* 26.5 (2015), pp. 889–902. DOI: 10.1109/TNNLS.2014.2327636.

[264] The Ocean Cleanup. https://theoceancleanup.com/. Accessed: 2024-03-25. 2024.

[265] The Rozalia project. https://www.rozaliaproject.org. Accessed: 2024-03-25. 2024.

[266] The SeaCleaners. https://www.theseacleaners.org. Accessed: 2024-03-25. 2024.

[267] J. Tian, W. Cheung, N. Glaser, Y.-C. Liu, and Z. Kira. "UNO: uncertainty-aware noisy-or multimodal fusion for unanticipated input degradation". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5716–5723. DOI: 10.1109/ICRA40945.2020.9197266.

[268] S. Tripathi, Z. C. Lipton, S. Belongie, and T. Nguyen. "Context matters: refining object detection in video with recurrent neural networks". In: *British Machine Vision Conference (BMVC)*. 2016. URL: http://vision.cornell.edu/se3/wp-content/uploads/2016/07/video_object_detection_BMVC.pdf.

[269] E. P. M. Troquay, A. Ilioudi, K. ter Burg, A. Dabiri, and B. De Schutter. "Enhancing underwater robotics computer vision models and environmental generalization for autonomous marine litter detection". In: *IEEE Journal of Oceanic Engineering* (Apr. 2024). under review.

[270] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders. "Selective search for object recognition". In: *Int. J. Comput. Vision* 104.2 (Sept. 2013), pp. 154–171. ISSN: 0920-5691. DOI: 10.1007/s11263-013-0620-5.

[271] A. Ullah, K. Muhammad, W. Ding, V. Palade, I. U. Haq, and S. W. Baik. "Efficient activity recognition using lightweight CNN and DS-GRU network for surveillance applications". In: *Applied Soft Computing* 103.107102 (2021). ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2021.107102.

[272] UNEP. *From pollution to solution: a global assessment of marine litter and plastic pollution*. UNEP report UNEA process. 2021, p. 148. URL: https://wedocs.unep.org/%20bitstream/handle/20.500.11822/36963/POLSOL.pdf.

[273] Unity Technologies. *Unity game engine*. Version 2024.1. Unity Technologies. San Francisco, CA, 2024. URL: https://unity.com/.

[274] Unity technologies. *Camera.ViewportToWorldPoint*. Accessed October 31, 2024. 2024. URL: https://docs.unity3d.com/ScriptReference/Camera.ViewportToWorldPoint.html.

[275] M. Valdenegro-Toro. "Submerged marine debris detection with autonomous underwater vehicles". In: *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. 2016, pp. 1–7. DOI: 10.1109/RAHA.2016.7931907.

[276] M. Valk. *Developing a robust and scalable plastic object detection model for The Ocean Cleanup*. Thesis MSc Business Analytics. Apr. 2021, p. 48–50.

[277] E. van Sebille, C. Wilcox, L. Lebreton, N. Maximenko, B. D. Hardesty, J. A. van Franeker, M. Eriksen, D. Siegel, F. Galgani, and K. L. Law. "A global inventory of small floating plastic debris". In: *Environmental Research Letters* 10.12 (Dec. 2015), p. 124006. ISSN: 1748-9326. DOI: 10.1088/1748-9326/10/12/124006.

[278] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[279] R. Verschae and J. Ruiz-del-Solar. “Object detection: current and future directions”. In: *Frontiers in Robotics and AI* 2 (2015), p. 29. ISSN: 2296-9144. DOI: 10.3389/frobt.2015.00029.

[280] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, M. Walczak, J. Garcke, C. Bauckhage, and J. Schuecker. “Informed machine learning – a taxonomy and survey of integrating prior knowledge into learning systems”. In: *IEEE Transactions on Knowledge and Data Engineering* 35.1 (2023), pp. 614–633. DOI: 10.1109/TKDE.2021.3079836.

[281] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. “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)*. 2023, pp. 7464–7475. DOI: 10.1109/CVPR52729.2023.00721.

[282] J. Wang, E. Sezener, D. Budden, M. Hutter, and J. Veness. “A combinatorial perspective on transfer learning”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[283] T. Wang, H.-J. Joo, S. Song, W. Hu, C. Keplinger, and M. Sitti. “A versatile jellyfish-like robotic platform for effective underwater propulsion and manipulation”. In: *Science Advances* 9.15 (2023), eadg0292. DOI: 10.1126/sciadv.adg0292.

[284] Y. Wang, Z. Xu, X. Wang, C. Shen, B. Cheng, H. Shen, and H. Xia. “End-to-end video instance segmentation with transformers”. In: *CoRR* abs/2011.14503 (2020). arXiv: 2011.14503. URL: https://arxiv.org/abs/2011.14503.

[285] Y. Wang, N. Li, Z. Li, Z. Gu, H. Zheng, B. Zheng, and M. Sun. “An imaging-inspired no-reference underwater color image quality assessment metric”. In: *Computers & Electrical Engineering* 70 (2018), pp. 904–913. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2017.12.006.

[286] J.-I. Watanabe, Y. Shao, and N. Miura. “Underwater and airborne monitoring of marine ecosystems and debris”. In: *Journal of Applied Remote Sensing* 13.4 (2019), p. 044509. DOI: 10.1117/1.JRS.13.044509.

[287] WEF. *The new plastics economy — rethinking the future of plastics*. Tech. rep. World Economic Forum, Ellen MacArthur Foundation and McKinsey & Company, 2016.

[288] K. Weiss, T. M. Khoshgoftaar, and D. Wang. “A survey of transfer learning”. In: *Journal of Big Data* 3.1 (2016), pp. 1–40. DOI: 10.1186/s40537-016-0043-6.

[289] J. Weston, D. Tolić, and I. Palunko. "Application of Hamilton-Jacobi-Bellman equation/Pontryagin's principle for constrained optimal control". In: *J. Optim. Theory Appl.* 200.2 (2024), pp. 437–462. DOI: 10.1007/S10957-023-02364-4.

[290] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. "Integrating physics-based modeling with machine learning: a survey". In: *ArXiv* abs/2003.04919 (2020).

[291] M. Wolf, K. van den Berg, S. P. Garaba, N. Gnann, K. Sattler, F. Stahl, and O. Zielinski. "Machine learning for aquatic plastic litter detection, classification and quantification (APLASTIC-Q)". In: *Environmental Research Letters* 15.11 (Nov. 2020), p. 114042. ISSN: 1748-9326. DOI: 10.1088/1748-9326/abbd01.

[292] World Economic Forum and Ellen MacArthur Foundation and McKinsey & Company. *The new plastics economy: rethinking the future of plastics*. [Online]. Available: https://www.ellenmacarthurfoundation.org/the-new-plastics-economy-rethinking-the-future-of-plastics. 2016.

[293] H. Wu, Y. Chen, N. Wang, and Z. Zhang. "Sequence level semantics aggregation for video object detection". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 9216–9224.

[294] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. "Galileo: perceiving physical object properties by integrating a physics engine with deep learning". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/d09bf41544a3365a46c9077ebb5e35c3-Paper.pdf.

[295] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[296] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. "Unseen object instance segmentation for robotic environments". In: *CoRR* abs/2007.08073 (2020). arXiv: 2007.08073. URL: https://arxiv.org/abs/2007.08073.

[297] S. Xu, M. Zhang, W. Song, H. Mei, Q. He, and A. Liotta. "A systematic review and analysis of deep learning-based underwater object detection". In: *Neurocomputing* 527 (Mar. 2023), pp. 204–232. DOI: 10.1016/j.neucom.2023.01.056.

[298] R. R. Yager. "On the dempster-shafer framework and new combination rules". In: *Information Sciences* 41.2 (1987), pp. 93–137. ISSN: 0020-0255. DOI: https://doi.org/10.1016/0020-0255(87)90007-7.

[299] L. Yang, Y. Fan, and N. Xu. "Video instance segmentation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 5187–5196. DOI: 10.1109/ICCV.2019.00529.

[300] M. Yang and A. Sowmya. "An underwater color image quality evaluation metric". In: *IEEE Transactions on Image Processing* 24.12 (2015), pp. 6062–6071. DOI: 10.1109/TIP.2015.2491020.

[301] S. Yang, X. Yu, and Y. Zhou. “LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example”. In: *International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*. 2020. DOI: 10.1109/IWECAI50956.2020.00027.

[302] S. Yang, Y. Fang, X. Wang, Y. Li, C. Fang, Y. Shan, B. Feng, and W. Liu. “Crossover learning for fast online video instance segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 8043–8052.

[303] T. Yang and A. B. Chan. “Recurrent filter learning for visual tracking”. In: *arXiv e-prints*, arXiv:1708.03874 (Aug. 2017), arXiv:1708.03874. arXiv: 1708.03874 [cs.CV].

[304] B. Yousuf, Z. Lendek, and L. Buşoniu. “Exploration-based search for an unknown number of targets using a UAV”. In: *IFAC-PapersOnLine* 55.15 (2022), pp. 93–98.

[305] Y. Yu, X. Si, C. Hu, and J. Zhang. “A review of recurrent neural networks: LSTM cells and network architectures”. In: *Neural Computation* 31.7 (2019), pp. 1235–1270. DOI: 10.1162/neco_a_01199.

[306] D. Zhang, J. Han, G. Cheng, and M. H. Yang. “Weakly supervised object localization and detection: a survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3074313.

[307] W. Zhang, L. Dong, X. Pan, P. Zou, L. Qin, and W. Xu. “A survey of restoration and enhancement for underwater images”. In: *IEEE Access* 7 (2019), pp. 182259–182279. ISSN: 2169-3536. DOI: 10.1109/access.2019.2959560. URL: http://dx.doi.org/10.1109/ACCESS.2019.2959560.

[308] Y. Zhang, Z. Huang, C. Chen, X. Wu, S. Xie, H. Zhou, Y. Gou, L. Gu, and M. Ma. “A spiral-propulsion amphibious intelligent robot for land garbage cleaning and sea garbage cleaning”. In: *Journal of Marine Science and Engineering* 11.8 (2023). ISSN: 2077-1312. DOI: 10.3390/jmse11081482.

[309] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu. “Object detection with deep learning: a review”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (2019), pp. 3212–3232. DOI: 10.1109/TNNLS.2018.2876865.

[310] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. “Domain generalization: a survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2023), pp. 4396–4415. DOI: 10.1109/TPAMI.2022.3195549.

[311] H. Zhu, H. Wei, B. Li, X. Yuan, and N. Kehtarnavaz. “A review of video object detection: datasets, metrics and methods”. In: *Applied Sciences* 10 (Nov. 2020), p. 7834. DOI: 10.3390/app10217834.

[312] M. Zhu and M. Liu. “Mobile video object detection with temporally-aware feature maps”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5686–5695. DOI: 10.1109/CVPR.2018.00596.

[313] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. “Flow-guided feature aggregation for video object detection”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 408–417. DOI: 10.1109/ICCV.2017.52.

[314] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. “Deep feature flow for video recognition”. In: *CoRR* abs/1611.07715 (2016). arXiv: 1611.07715. URL: http://arxiv.org/abs/1611.07715.

[315] P. Zhuang, Y. Wang, and Y. Qiao. “Wildfish: a large benchmark for fish recognition in the wild”. In: *Proceedings of the 26th ACM international conference on Multimedia*. Seoul Republic of Korea: ACM, Oct. 2018, pp. 1301–1309. ISBN: 978-1-4503-5665-7. DOI: 10.1145/3240508.3240616.

[316] K. Zuiderveld. “Contrast limited adaptive histogram equalization”. In: *Graphics Gems*. Ed. by P. S. Heckbert. Academic Press, 1994, pp. 474–485. ISBN: 978-0-12-336156-1. DOI: https://doi.org/10.1016/B978-0-12-336156-1.50061-6.

ACKNOWLEDGEMENTS

My PhD has been an extraordinary journey of personal growth. This growth would not have been possible without the help and support of several people, to whom I am deeply grateful.

Firstly, I would like to thank my supervisors, Prof. Bart De Schutter and Dr. Azita Dabiri, for their guidance and support over the years. Bart gave me the opportunity to pursue my PhD in his group. His insights into critical thinking, research content and quality, and the practical aspects of my work greatly enriched my research. Collaborating with him was a rich learning experience, and I hope I did not disappoint. I am incredibly grateful for his understanding and support during a personally difficult time. Special thanks also to Azita, whose feedback and constant availability have greatly contributed to the success of this project.

I extend my gratitude to my PhD committee members Prof. Marcel J.T. Reinders, Prof. Robert Babuska, Dr. Matias A. Valdenegro-Toro, Prof. Lucian Busoniu, and Dr. Junaed Sattar, for their valuable feedback and the time invested in evaluating this dissertation.

I am also grateful to the people from the SeaClear consortium, with whom I collaborated most closely: Antun, Mihalis, Cosmin, Stefan, Vicko, Lucian, and Ivana. Thank you for the excellent collaboration, for always being so helpful and kind, and for the great moments we shared after the project reviews and demo preparations.

Within the Mechanical Engineering faculty, I would like to thank many people who supported me in different ways. Special thanks to Alina Colling, whose energy and motivation in the PhD Council were always inspiring. I also thank Mascha Toppenberg and Francy Ahaus-Smeets for their help with my administrative tasks. Special thanks to Michiel Haanschoten, from TU Delft's ICT support, who patiently helped me recover my Ubuntu laptop on numerous occasions after crashing it (almost monthly) with incompatible driver updates.

I am incredibly grateful to Leila, for her support and friendship since the early days of the PhD. Your inspiration and collaboration made this journey much more rewarding. Many thanks to Kaya for our wonderful and ongoing collaboration, and to David, Giorgos, Emilio, Sreeshma, Maria, and Abhimanyu for the memorable moments we shared. I also want to thank Suad for connecting everyone in the department with sports groups and competitions, and Clara for the great time we had while organizing the video competition with her and Suad. To my office mates, Sreeshma, Kaya, Maria, and Pedro, thank you for the stimulating discussions and the moments we spent together. Special thanks also to Mathieu, Mallika, and Eline for our collaborations during their Master theses. It was truly a pleasure working with you.

This thesis is dedicated to my family. To my parents, Eleni and Giannis, your

unconditional love, patience, and support have been guiding me throughout this journey. To my sister Vasiliki for the encouragement. To my dearest friend Christina, your incredible positive energy, encouragement, and support have meant more than words can say. And to my beloved partner, Michalis, thank you for standing by my side through every high and low. Your love and support have been a tremendous blessing throughout this PhD journey.

CURRICULUM VITAE



Athina Ilioudi obtained her engineering Diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki, Greece, in 2014. Her thesis focused on the control of the powertrain of an electric racing vehicle. Following her graduation, she worked as a research assistant at the Centre for Research and Technology Hellas (CERTH).

In 2018, she received a joint M.Sc. degree in Smart Electrical Networks and Systems from KTH Royal Institute of Technology, Stockholm, Sweden, and Eindhoven University of Technology, in the Netherlands. Her thesis focused on the design of a linear quadratic controller for trajectory tracking of a medical manipulator system. After graduation, she worked as a mechatronics designer at Philips, R&D Image Guided Therapy Systems.

In June 2020, she became a PhD Candidate at the Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology, under the supervision of Prof. dr. ir. Bart De Schutter and Dr. Azita Dabiri.

During her PhD project, she conducted research on underwater computer vision, image enhancement techniques, and physics-informed neural networks for computer vision applications. Her research interests include computer vision and robotics for underwater applications.

LIST OF PUBLICATIONS

7. A. Ilioudi, A. Dabiri, B. J. Wolf, and B. De Schutter. "Deep learning for object detection and segmentation in videos: Toward an integration with domain knowledge". In: *IEEE Access* 10 (2022), pp. 34562–34576. DOI: 10.1109/ACCESS.2022.3162827

6. A. Ilioudi, B. J. Wolf, A. Dabiri, and B. De Schutter. "Towards establishing an automated selection framework for underwater image enhancement methods". In: *OCEANS 2023 - Limerick*. 2023, pp. 1–6. DOI: 10.1109/OCEANSLimerick52467.2023.10244710

5. E. P. M. Troquay, A. Ilioudi, K. ter Burg, A. Dabiri, and B. De Schutter. "Enhancing underwater robotics computer vision models and environmental generalization for autonomous marine litter detection". In: *IEEE Journal of Oceanic Engineering* (Apr. 2024). under review

4. A. Đuraš, B. J. Wolf, A. Ilioudi, I. Palunko, and B. De Schutter. "A dataset for detection and segmentation of underwater marine debris in shallow waters". In: *Scientific Data* 11.1 (Aug. 2024), p. 921. ISSN: 2052-4463. DOI: 10.1038/s41597-024-03759-2

3. A. Ilioudi, S. Sosnowski, E. Banken, P. Bevanda, J. Brüdigam, L. Buşoniu, Y. Chardard, C. Delea, B. De Schutter, A. Đuraš, C. Hertel-ten Eikelder, S. Heshmati-Alamdari, V.-M. Maer, I. Palunko, I. Pozniak, V. Prkačin, and D. Tolić. "The SeaClear system: an intelligent multi-robot solution for autonomous cleanup of marine debris on the seabed". In: *Engineering Applications of Artificial Intelligence* (May 2024). Submitted, under review

2. A. Ilioudi, L. Gharavi, K. ter Burg, A. Dabiri, and B. De Schutter. "Deep learning and physics-based models fusion for classification in out-of-distribution scenarios". In: *Information Sciences* (2025). to be submitted

1. M. D'heer, A. Ilioudi, B. De Schutter, A. Dabiri, and A. Jamshidnejad. "Domain-informed neural networks for detecting underwater moving objects with noisy datasets". In: *Knowledge-Based Systems* (2025). to be submitted