

Martin Skaldebø

Visual Aided Deep Learning Applications for Underwater Operations

Master's thesis in Marine Technology

Supervisor: Ingrid Schjøtlberg

November 2019

Martin Skaldebø

Visual Aided Deep Learning Applications for Underwater Operations

Master's thesis in Marine Technology
Supervisor: Ingrid Schjølberg
November 2019

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology



NTNU – Trondheim
Norwegian University of
Science and Technology

Visual Aided Deep Learning Applications for Underwater Operations

Master Thesis

Martin Breivik Skaldebø

September 9, 2019

Supervisor: Ingrid Schjølberg

Task Description



NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

Master Thesis 2019

Martin Skaldebø

Visual Aided Deep Learning Applications for Underwater Operations

Work Description

Subsea panels represent the interface between ROV and subsea facility. Subsea panels consist of multiple interfaces of which each produce a different configuration outcome when manipulated. Correct detection and classification of subsea panel interfaces is imperative for autonomous subsea intervention missions. However, classifying based on classic computer vision (CV) is difficult and error-prone due to the requirement of manually created feature extraction. Deep learning methods could be an alternative, but then new problems occur. To train a framework at a real system can be highly time consuming and vast amounts of data is required to train in a simulated environment. Training solely in simulated situations are error-prone due to a gap between the domains, and there is no guarantee that simulated results will transfer to the real world. This project seeks to investigate deep learning methods to transfer knowledge between different domains and in this way decrease the reality gap. In addition, deep learning methods for object detection using simple monocular camera will be investigated, and the possibility to incorporate such systems with dynamic positioning (DP) systems for underwater robots.

Scope of work

Resume:

- Review relevant literature related to machine learning in underwater operations
- Review relevant literature related to transfer learning and object detection using deep learning

Paper one

- Investigate the use and adaptability of domain transfer methods in underwater environments
- Obtain datasets from different underwater environments
- Apply the most relevant method to the relevant datasets
- Investigate the feasibility of further development of such methods in the relevant environments

Paper two

- Perform object detection of a known object using deep learning methods
- Generate and label dataset of the relevant object to be detected
- Design and incorporate a dynamic positioning (DP) system with the detector in order to perform DP relative to the known object

Trondheim, 08.08.19

Ingrid Schjølberg

Preface

This thesis is submitted as the final part of the master of science program at the Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis is delivered during the fall 2019 and the work was conducted at the Department of Marine Technology, NTNU. Professor Ingrid Schjølberg has been the main supervisor for this thesis.

The thesis is assembled with a collection of two appended scientific articles and an associated resume as main report. The collective topic for the thesis concerns vision based deep learning systems and how such systems can aid in increasing autonomy in underwater operations. The two appended articles are *Transfer Learning in Underwater Operations* and *Dynamic Position of an Underwater Vehicle using Monocular Vision-Based Object Detection with Machine Learning*.

Abstract

The 4th industrial revolution has been ongoing the last four years and results are starting to show in the market. Google constantly reveals new technology that can beat humans at different strategy games and they have developed AIs that can learn to walk on their own. Also, governments and classification companies continuously works towards a set of regulations and guidelines for autonomous ships, Yara Birkeland is a ship that is planned to be released in 2020 that will move towards fully autonomous operations within 2022. Autonomy in underwater operations are also experiencing a growth which shows in the expected market growth.

Machine learning is believed to play an important role in the shift towards autonomy. Feature extraction methods for subsea applications existing today mainly consist of acoustic sensors. Feature extraction using camera vision are rarely used, but improvements within neural networks, especially convolutional neural networks (CNN), have proved promising results. This thesis will investigate the use of such methods for subsea applications. The thesis specifically highlights two different objectives, that are dispersed at two appended articles.

The first objective concerns the reality gap, which describes the phenomena that occurs when transferring knowledge between domains, specifically the artificial simulated domain and the real world domain. Existing methods focusing on this have been explored and compared in order to generate an outline of existing solutions. Due to the harsh environment in subsea applications, the use of camera vision for feature extraction has been shallow. Bad lighting, marine snow, constantly change of environment etc. makes it difficult to model the subsea environment. Thus, there will almost certainly exist a gap between a simulated environment and the real world. In the last few years numerous frameworks targeting domain transfer have been published. Generative Adversarial Networks (GAN) has stood out as maybe one of the recent most promising methods regarding domain transfer. This thesis investigate the use of a framework called `CycleGAN` which introduces a cycle consistent feature in the ordinary GAN. `CycleGAN` has proven good results in several disciplines and has in this thesis been tested for the first time for datasets regarding an underwater environment. The method consist of taking an image as input and generating a similar image in another domain, e.g. taking an image of a rendered subsea panel as input and generating a real-looking subsea panel in a real underwater environment as output. In this thesis the framework is applied on two different underwater image datasets and is proven to accomplish a good mapping between simulated and real domains.

The second objective concerns visual based object detection using monocular camera. Deep learning methods for object detection are investigated in collaboration with methods for extracting relevant features from the respective detectors in order to aid a dynamic positioning (DP) system. A detector based on `YOLOv3` is applied on a dataset collected from the Marine Cybernetics Laboratory at NTNU. The dataset includes a known object in the laboratory pool environment. Once the detector is trained and successfully detects the relevant object, a spatial scaling function and a DP system for the underwater vehicle `BlueROV2` is designed. The `BlueROV2` performs DP relative to the object extracting position information and spatial features from the detector. The system performs well and the `BlueROV2` manages to perform DP in a good manner with correct extraction of both localization and spatial features of the object of interest.

Sammendrag

Den 4. industrielle revolusjonen har pågått de siste fire årene og resultater har begynt å vise seg i industrimarkedet. Google slipper stadig ny teknologi som kan slå mennesker i et mangfold av strategispill og de har utviklet kunstig intelligens som kan lære å gå på egenhånd. Statlige instanser og klassifikasjonsselskaper samarbeider mot regler og retningslinjer for autonome skip, Yara Birkeland er et skip som er planlagt ferdig i 2020 som vil gå mot full autonom operasjon innen 2022. Autonomi i undervannsoperasjoner erfarer også vekst som vises igjen i økning i markedet.

Maskinlæring er forventet å være en bidragsyter i skiftet mot en mer autonom fremtid. Metoder for innhenting av relevant informasjon i undervannsapplikasjoner som eksisterer idag bruker hovedsakelig akkustiske sensorer. Metoder for informasjonsinnhenting som bruker kamerabilder er sjeldent brukt, men forbedringer innen nevralt nettverk, spesielt Convolutional Neural Networks (CNN), har vist lovende resultater. Denne rapporten vil undersøke bruken av slike metoder for undervannsoperasjoner. Rapporten trekker spesielt frem to problemstillinger, som er fordelt på to vedlagte artikler.

Den første problemstillingen anngår realitetsgapet, som beskriver fenomenet som oppstår når kunnskap blir overført mellom domener, mer presist mellom det kunstige simuleringsdomenet og den virkelige verden. Eksisterende metoder angående dette har blitt undersøkt og sammenlignet for å danne et utsnitt av eksisterende metoder. Grunnet det tøffe miljøet under vann har bruken av kamerabilder til informasjonsinnhenting vært liten. Dårlig belysning, maritim snø, et miljø i kontinuerlig endring etc. gjør det vanskelig å modellere et undervannsmiljø. Det vil derfor nesten garantert eksistere et realitetsgap mellom det simulerte domenet og den virkelige verden. I løpet av de siste årene har flere rammeverk som angriper realitetsgapet oppstått og blitt publisert. Generative Adversarial Networks (GAN) har muligens stått frem som en av de mest lovende metodene for domenetransformasjoner de siste årene. Denne rapporten undersøker en metode av dette rammeverket kalt CycleGAN, som introduserer en sykelkonsistens egenskap i det originale GAN rammeverket. CycleGAN har vist gode resultater inne flere disipliner og har i denne rapporten for første gang blitt anvendt på undervannsapplikasjoner. Metoden består av å ta et bilde som input og reprodusere det samme bildet i et annet domene som output, e.g. ta et generert bilde fra simuleringsdata av et undervannspanel som input og produsere et virkelighetsnært bilde av det samme undervannspanelet. I denne rapporten er rammeverket anvendt på to forskjellige dataset med bilder fra undervannsmiljøer og en god kartlegging mellom simuleringsbaserte domener og den virkelige verden er oppnådd.

Den andre problemstillingen omhandler optisk basert objekt detektering ved bruk av monokularkamera. Deep learning metoder for objekt detektering er undersøkt i unison med metoder for å ekstrahere relevante egenskaper fra de respektive detektorene for å bidra i et dynamisk posisjonerings (DP) system. En detekteringsalgoritme basert på YOLOv3 er anvendt på et dataset sammensatt av bilder fra marin kybernetisk laboratoriet (MC-lab) ved NTNU. Datasettet inkluderer bilder av et kjent objekt i bassenget i laboratoriet. Når detekteringsalgoritmen er ferdig trent og kan vellykket detektere det relevante objektet, er en skaleringsfunksjon samt et DP-system for undervannsdronen BlueROV utviklet. BlueROV2 utfører DP relativ til det kjente objektet ved å innhente relevant informasjon angående posisjon og avstander ved hjelp av detekteringsalgoritmen. Systemet fungerer tilfredsstillende og klarer å utføre DP med korrekt informasjonsinnhenting av både posisjon og avstander til objektet fra kamerabilder.

Aknowledgements

I am deeply grateful for the help and guidance I have received from my supervisor Prof. Ingrid Schjølberg. She has always been positive and supportive and encouraged me to pursue my fields of interest. She created an impressive cooperation between all her masters students, PhD students and post doc fellows. This provided an open environment where people supported each other and help was always available. I would like to thank Bent O. Arnesen Haugaløkken and Albert Sans Muntadas for considerable help and collaboration with the articles. The work produced from the articles would not be possible without their exceptional guidance and teamwork. I would also like to thank Mikkel C. Nielsen and Jeevith Hedge for help and guidance whenever I have had any issues regarding research topics or implementation.

Finally I would thank fellow students, friends and roommates for making the last 5 years as a student in Trondheim truly unforgettable. The memories and friendships I have acquired is something I will treasure for the rest of my life.

Martin B. Skaldebø
Trondheim, August 13th 2019

Contents

| | |
|---|-------------|
| Preface | iii |
| Abstract | iv |
| Sammendrag | v |
| Acknowledgements | vi |
| Contents | viii |
| List of Figures | ix |
| List of Tables | x |
| Nomenclature | xi |
| | |
| I Main Report | 1 |
| | |
| 1 Introduction | 2 |
| 1.1 Motivation | 2 |
| 1.2 Underwater operations today | 3 |
| 1.3 Towards autonomy | 3 |
| 1.3.1 Technological modules of autonomous systems | 6 |
| 1.4 Objectives | 7 |
| 1.5 Main Contributions | 7 |
| 1.6 Outline of the Thesis | 8 |
| | |
| 2 Facilities and Equipment | 9 |
| 2.1 Facilities | 9 |
| 2.1.1 Marine Cybernetics Laboratory | 9 |
| 2.1.2 Trondheimsfjorden | 10 |
| 2.2 Equipment and Software | 10 |
| 2.2.1 BlueRov2 | 10 |
| 2.2.2 Robotic Operating System (ROS) | 11 |
| 2.2.3 Python | 11 |
| 2.2.4 Tensorflow | 11 |
| 2.2.5 PyTorch | 12 |
| | |
| 3 Background Theory | 13 |
| 3.1 Neural Networks | 13 |
| 3.1.1 Training a Neural Network | 15 |
| 3.2 Convolutional Neural Networks (CNN) | 18 |
| 3.2.1 Convolution Layer | 18 |

| | | |
|-----------|---|-----------|
| 3.2.2 | Pooling Layer | 20 |
| 3.2.3 | Fully Connected Layer | 21 |
| 3.3 | Machine Learning | 22 |
| 3.3.1 | Learning Algorithms | 22 |
| 3.3.2 | Deep Learning | 24 |
| 3.4 | Vision Based Learning | 25 |
| 3.4.1 | Domain Transfer | 25 |
| 3.4.2 | Object Detection | 25 |
| 4 | Methodology | 27 |
| 4.1 | Autoencoders | 27 |
| 4.2 | Generative Adversarial Networks (GAN) | 28 |
| 4.2.1 | CycleGAN | 29 |
| 4.3 | Object Detection | 30 |
| 5 | Conclusions and Further Work | 33 |
| | Bibliography | 33 |
| II | Collection of Articles | 38 |
| | Article 1: Transfer Learning in Underwater Operations | 39 |
| | Article 2: Dynamic Positioning of and Underwater Vehicle using Monocular Vision-Based Object Detection with Machine Learning | 48 |

List of Figures

| | | |
|------|--|----|
| 1.1 | Worlds largest listed companies by market capitalization [7] | 4 |
| 1.2 | Industrial revolutions [10] | 5 |
| 1.3 | 5 modules of autonomous systems [20, 21, 22, 23, 24] | 6 |
| 2.1 | MC-lab. | 9 |
| 2.2 | Location of Trondheimsfjorden. | 10 |
| 2.3 | BlueROV2 underwater vehicle | 11 |
| 3.1 | Simple illustration of a graph | 14 |
| 3.2 | Simple Neural Network | 14 |
| 3.3 | Weight w_{24}^3 in a neural network | 15 |
| 3.4 | Optimal gradient | 17 |
| 3.5 | The saddle point problem | 17 |
| 3.6 | Convolution in matrix perspective [41] | 19 |
| 3.7 | Edge detection applied to a black and white picture | 20 |
| 3.8 | Edge detection applied to a real picture [42] | 20 |
| 3.9 | Max- and average-pooling | 21 |
| 3.10 | Fully connected CNN [43] | 21 |
| 3.11 | Trends of <i>machine learning</i> the last 10 years | 22 |
| 3.12 | Supervised learning | 23 |
| 3.13 | Unsupervised learning | 23 |
| 3.14 | Reinforcement learning | 24 |
| 3.15 | Ordinary network vs deep network | 24 |
| 3.16 | Object Detection Methodologies | 26 |
| 3.17 | Machine Learning methodologies | 26 |
| 4.1 | Architecture of autoencoders | 27 |
| 4.2 | Architecture of the CycleGAN method taken from [62] | 29 |
| 4.3 | Published results from [62]. | 30 |

List of Tables

| | | |
|-----|--|----|
| 1.1 | Levels of autonomy defined in [16] | 6 |
| 2.1 | Bluerov2 specifications | 11 |
| 3.1 | Commonly used activation functions | 16 |
| 4.1 | Darknet-53 | 32 |

Nomenclature

List of Abbreviations

| | |
|--------|--|
| AI | Artificial Intelligence |
| ALP | Algorithmic Natural Language |
| ANN | Artificial Neural Network |
| AUV | Autonomous Underwater Vehicle |
| CAGR | Compound Annual Growth Rate |
| cGAN | Conditional Generative Adversarial Network |
| CNN | Convolutional Neural Network |
| CoGAN | Coupled Generative Adversarial Network |
| CPU | Central Processing Unit |
| DFPN | Deconvolutional Feature Pyramid Network |
| DOF | Degrees of Freedom |
| DP | Dynamic Positioning |
| GAN | Generative Adversarial Network |
| GNSS | Global Navigation Satellite System |
| GPU | Graphics Processing Unit |
| HMI | Human-Machine-Interface |
| IaaS | Infrastructure as a Service |
| IMR | Inspection, Maintenance and Operation |
| IT | Information Technology |
| MC-lab | Marine Cybernetics Laboratory |
| MSE | Mean Squared Error |
| NLP | Natural Language Processing |
| OSI | Open Source Initiative |
| PaaS | Platform as a Service |
| PSO | Particle Swarm Optimization |
| ReLU | Rectified Linear units |

| | |
|------|---------------------------|
| RGB | Red Green Blue |
| ROS | Robotic Operating System |
| ROV | Remotely Operated Vehicle |
| RPN | Regional Proposal Network |
| SaaS | Software as a Service |
| SVM | Support Vector Machine |
| YOLO | You Only Look Once |

Part I

Main Report

Chapter 1

Introduction

This chapter gives a short introduction and describes the motivation behind the thesis. How underwater operations are conducted today will also be discussed as well as how the industry is moving towards a more autonomous future. General contributions and outline of the thesis are also given.

1.1 Motivation

The underwater robotic market size is claimed to reach USD 6.74 Billion by 2025 [1]. This corresponds to a Compound Annual Growth Rate (CAGR) of 13.5%. By comparison, Apple Inc. had a CAGR of 9.2% the last 5 years, per July 30th 2019 [2]. The same report predicts that autonomous underwater vehicles (AUV) will account for USD 1.48 billion by 2025. The Norwegian Government is investing in the ocean space when designing the concept *Ocean Space Center*. The concept has a planned investment of 4.7 billion NOK [3].

Activity, interest and growth within the ocean space are accordingly unquestionable. And with such growth, advancement in the technology is forthcoming. In the last years, machine learning has experienced a substantial growth in both media coverage and technological applications and like most industries, underwater operations experience changes towards more autonomous systems. The wide interest and will to achieve progress that is shown today generates motivation for further investments in the field. Machine learning is believed to play a significant role in this shift towards autonomy.

Teleoperated systems existing in inspection, maintenance and repair (IMR) operations today, the human operator is aided by visual and sensory feedback in order to assess the situation, make decisions and remotely execute tasks. Making such systems more autonomously increases the demands regarding the sensory systems and implemented software. What concerns sensors in underwater operations, acoustic sonars have for a long time been preferable. However, recent technological advance within camera systems and the use of visual aid proves that camera systems have potential to be the preferable systems for short range navigation. Moreover, visual aided systems may provide systems with higher spatial and temporal resolutions than the acoustic counterpart [4]. Nonetheless, it is not straightforward to use camera systems in underwater environments, especially when paired with robotic systems during semi- or fully autonomous operations. The underwater scene is considered one of the most difficult conditions to perform optical detection and recognition of features and patterns, partly because underwater image quality heavily depends on absorption and scattering of light [5, 6].

Despite the obstacles regarding machine learning, the interest for such applications in underwater environments is quite understandable. The underwater environment is exposed to constantly changing environments - marine snow, bad lighting etc. This provides extremely high complexity for modeled solutions. In order to model such uncertain environments it would require vast amounts of information about the system. Machine learning and neural networks provides an alternative solution to this, and

can help ensure that autonomous systems can cope with the high uncertainty present in underwater environments. Moreover, technological advances and lowered costs of graphical processing units and cameras provides a future for visual aided sensors. Simultaneously, the same type of development has been seen in commercial underwater vehicle products, such as the BlueROV2, which allows customization and testing of new hardware with user-made software. Incorporating visual aided tools for underwater vehicles enables more autonomous functionality in underwater robotics, such as tracking of objects during IMR or visually aided manipulation operations, whether it is used in exploration operations, within the marine oil and gas or the aquaculture industry.

Although the existing frameworks have been tested and indicates promising results, they have never, for my knowledge at this time, been applied in underwater conditions. The underwater environment is a harsh and unforgiving environment and there could be elements here that will generate problems in the existing solutions. Conditions at the bottom of the sea includes total darkness, sediments clouding the view, currents and other factors which complicates operations. It is not straight forward to apply something that works for one specific task to a complete different problem. Thus, it is not guaranteed that the existing methods will provide satisfying results for the problems this thesis will investigate.

1.2 Underwater operations today

Underwater operations today are highly dependant on human operators. Operations previously executed by human divers are now mostly transferred to remotely operated vehicles (ROV). This has enabled safer operations and the use of human divers for offshore applications have been gradually faded out. Now the industry is experiencing a new shift towards more autonomous operations were ROVs will become more independent of human operators. Performing operations on the seabed today is a costly affair. The ROV needs to be supported by an operating team that controls it. A mothership is also necessary which brings the requirement of a crew as well. Thus, in order to execute an underwater operation a mothership must be operated along with pay for the crew and ROV team. Such an operation is not only costly, but prone to human error from the operator, highly weather dependant due to the mothership at surface level and spatial restricted due to the umbilical of the underwater vehicle.

AUVs are also used. Since AUVs brings a higher level of autonomy they circumvents some of the issues with ROVs. Especially regarding umbilical and constant supervision from crew and mothership. However, with higher level of autonomy comes new requirements regarding autonomous complexity. AUVs are vulnerable to loss of vehicle and data, and has less power storage than the ROV. Since global navigation satellite system (GNSS) measurements are not applicable underwater, vehicle operation in this domain lacks localization measurements and are prone to accumulation of localization error. Today, the most common measurements and signal data arrives from acoustic sensors. Such signals are prone to data loss due to transmission losses, acoustic noise in thrusters, signal reflections on different surfaces, absorption loss and more. Feature extraction using camera vision are rarely used, but improvements within artificial neural networks (ANN), especially convolutional neural networks (CNN), shows promising results. In the presented work, systems using visual aid will be investigated. This is mostly motivated by the rapid advance withing CNN and other computer vision frameworks building on CNN.

1.3 Towards autonomy

At August 31st 1910, at the Dedication of the John Brown Memorial Park in Osawatomie, Kansas, Theodore Roosevelt delivered a famous speech [7]. During the speech the former president expressed concerns regarding the American economy. Roosevelt said,

”The absence of effective State, and, especially, national, restraint upon unfair money-getting has tended to create a small class of enormously wealthy and economically powerful men, whose chief object is to hold and increase their power.”

He was concerned a handful corporate giants would generate enormously wealth compared to the common man, which could lead to increase in political influence and power. Roosevelt’s words are as relevant today as in the late summer of 1910. A handful of giant companies rises again as the sole rulers of the global market. Apple, Alphabet, Microsoft and the companies next in line dominate today’s economy just as surely as US Steel, Standard Oil and Sears and Roebuck and Company dominated the economy of Roosevelt’s days.

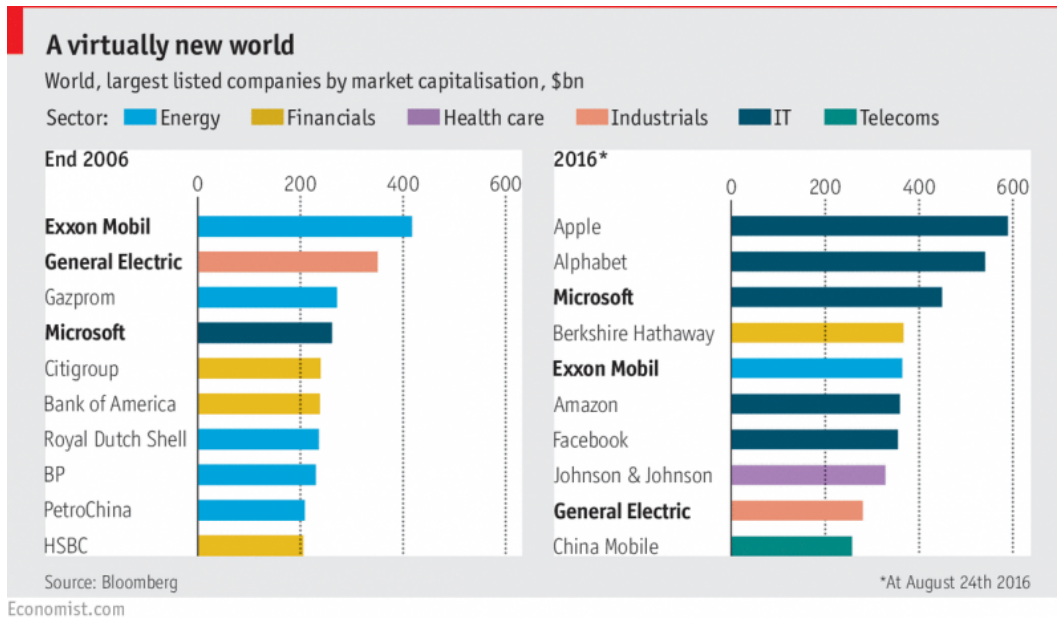


Figure 1.1: Worlds largest listed companies by market capitalization [7]

Figure 1.1 shows how only three companies that dominated the economy in 2006 remained on top in 2016. Another important matter taken from the figure is the number of information technology (IT) companies, from only one in 2006 to five out of ten in 2016. Per November 16th 2018, both Apple and Amazon have reach a market value above \$1 trillion [8]. Roosevelt’s concerns were in other words genuine.

These IT companies are the lighthouse in the 4th industrial revolution we are a part of today. This revolution is also referred to as Industry 4.0 [9]. Figure 1.2 illustrates the four industrial revolutions.

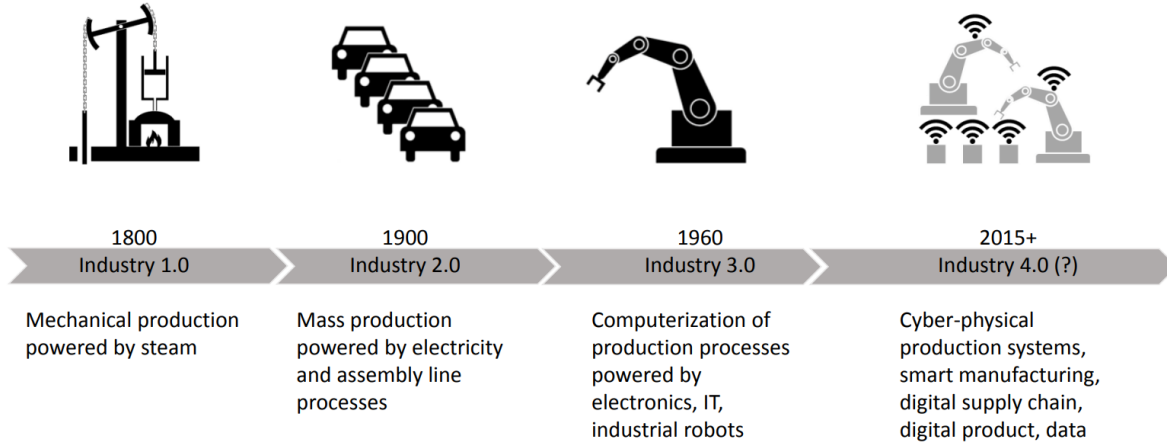


Figure 1.2: Industrial revolutions [10]

The first industrial revolution consisted of steam and machines that mechanized the work of our ancestors. Next was electricity bringing assembly lines and mass production around Roosevelt's time. Then the advent of computerization and the beginning of automation which began replacing the human factor of the assembly lines. And now we are in the fourth revolution where computers and automation will join forces in a revolutionary way. Systems equipped with machine learning algorithms will provide controls and robotics operating with minimal input from humans. [11] defines autonomous systems as a system that possesses self-governing characteristics, which allows it to perform pre-specified tasks without human intervention. However, it is difficult to determine if a system is autonomous or not with a classic binary value. Therefore it is more customary to talk about the level of autonomy. It is hard to define a general set of levels of autonomy because it should necessarily be adapted to different situations. [12] defines 6 levels of autonomy for autonomous cars, while [13, 14] has defines 10 levels of autonomy for aerial vehicles. [15] has defined 4 levels of autonomy for aerial vehicle which later has been adopted by [16] for marine applications. This definition is considered the most relevant for the problem addressed in this thesis and are presented in Table 1.1

Table 1.1: Levels of autonomy defined in [16]

| Level of autonomy | Description |
|---|---|
| 1. Automatic operation (Remote control) | The system operates automatically. The human operator directs and controls all high-level mission planning functions, often preprogrammed. System states, environmental conditions and sensor data are presented to the operator through a human-machine-interface (HMI) (human-in-the-loop/human operated) |
| 2. Managements by Consent | The system automatically makes recommendations for mission or process actions related to specific functions, and the system prompts the human operator at important points in time for information or decisions. At this level the system may have limited communication bandwidth, including time delay, due to, e.g. distance. The system can perform many functions independently of human control when delegated to do so (human-delegated) |
| 3. Management by Exception | The system automatically executes mission-related functions when response times are too short for human intervention. The human may override or change parameters and cancel or redirect action within defined time lines. The operator's attention is only brought to exceptions for certain decisions (human-supervisory control) |
| 4. Highly Autonomous operation | The system automatically executes mission or process related function in an unstructured environment with ability to plan and re-plan the mission or process. The human may be informed about the progress. The system is independent and "intelligent" (human-out-of-the-loop) |

Reaching full autonomy is not just a question about the technological challenges, but also the political and social aspects. Yara Birkeland is planned to be the world's first fully electric and autonomous container ship with zero emissions [17]. The ship will be delivered in 2020 and will move towards fully autonomous operations in 2022. To succeed with such a project several participants have contributed. Kongsberg Maritime is responsible for technological solutions, the classification company DNV-GL provides new guidelines for autonomous ships [18] and the Norwegian Maritime Authority provides a new set of rules and regulations [19]. For this thesis it is mostly the technological aspects that will be regarded.

1.3.1 Technological modules of autonomous systems

An autonomous system can be said to be composed of five modules [11] illustrated in Figure 1.3.



Figure 1.3: 5 modules of autonomous systems [20, 21, 22, 23, 24]

The modules are

- **Sensors:** As humans use smells, vision and hearing to sense the world around them, an autonomous system needs sensors in order to understand the environment it operates in.
- **Learning:** An autonomous system needs to learn from its environment and actions. In order for a system to act independently from a human operator, it needs to adjust its actions relative to the feedback it gets from its sensors. Machine learning algorithms can also be applied in order for the system to solve tasks it had no prior knowledge to.
- **Planning:** When the system should conduct a task, it first needs a plan. If a robot should relocate from position A to B, it needs a path to follow in order to avoid eventual obstacles.
- **Diagnosis:** An autonomous system needs to constantly run diagnosis. If it has low battery level it needs to recharge, if it can't complete a task due to physical constraints it has to communicate this to other parties. The system has to constantly check these along with other factors in order to be operational.
- **Control:** The system has to be able to control the specific task(s) it is made for. A robot operating an assembly line needs actuators to move objects. An underwater vehicle needs thrusters in order to control its position and movements and actuators to perform tasks as turning switches on subsea panels.

For this thesis all modules will be considered. DeepMind learned an artificial intelligence (AI) with no prior knowledge of walking to stand up, walk, run and jump [25], using machine learning algorithms. This proves how such algorithms can mimic the human brain regarding learning. The AI had no prior knowledge about walking, it was only given rewards if it managed to do tasks such as move in a certain direction. Neural networks are widely used within machine learning algorithms in order to mimic the brains behaviour. Deepmind has also learned an AI to play the game GO with the use of convolutional neural networks [26].

1.4 Objectives

This thesis investigates vision based methods for deep learning applications in underwater operations. Feasibility, as well as limitations and challenges, of existing methods are discussed. Due to insufficient knowledge about the application at underwater operations and unacquainted results, only the most promising of the existing methods are implemented and tested in the appended articles. This is to ensure that the methods are transferable to dissimilar training sets before devoting resources on bad prospects. Relevant topics that are considered in the thesis are

1. The role of machine learning in the shift towards more autonomous systems
2. State of the art transfer learning methods
3. State of the art object detection methods
4. The role of vision based deep learning in the increasingly autonomy industry
5. Adequacy of existing methods in underwater environments

1.5 Main Contributions

The main contributions for this thesis are applications of machine learning frameworks in underwater environments. There are two appended papers, each with different applications, with individual contributions summarized below. Both papers are submitted to conferences for publication.

Paper 1 investigates a method for transfer learning in vision-based underwater operations. In the presented work, experiments are conducted for two different datasets obtained in an underwater environment. The paper provides a collective overview of state-of-the-art frameworks targeting transfer learning topics. Moreover, suggests solutions for reduction of the reality gap in the learning process of machines. The paper is submitted to the *Oceans 2019 Marseille Conference & Exhibition*.

Paper 2 includes the design of a DP-system where the vehicle will have a desired position relative to a known object, where position of the object is extracted from monocular camera. The object detection scheme is provided with a labelled machine learning application, and an effective labeling scheme is designed in order to cope with labeling of large datasets. The paper is submitted to the *Oceans 2019 Seattle Conference & Exposition*.

1.6 Outline of the Thesis

The next chapters will provide necessary background theory and insight relevant for the appended papers. The general outline of the thesis is as follows.

Chapter 1 describes the motivation behind this thesis and how underwater operations are developing towards autonomy, as well as the main objectives and contributions of the thesis.

Chapter 2 describes the facilities and equipment used in simulations and experimental testing.

Chapter 3 describes background theory regarding how neural networks and machine learning can aid in a more autonomous direction. Existing solutions for similar situations are also presented.

Chapter 4 presents the methodology and frameworks utilized in the papers.

Chapter 5 provides conclusions and recommendations for further work.

Chapter 2

Facilities and Equipment

This chapter considers the facilities and equipment used in simulations and experimental testing. Important software are also described.

2.1 Facilities

This section provides information about the relevant facilities and locations that are used in simulations and experimental testing.

2.1.1 Marine Cybernetics Laboratory

The Marine Cybernetics Laboratory (MC-lab) is a laboratory located at the Department of Marine Technology, NTNU, in Trondheim. The laboratory consist of a control room and a towing tank with dimensions $40\text{m} \times 6.46\text{m} \times 1.5\text{m}$ and includes a wave maker, a towing carriage and a 6 degrees of freedom (DOF) real-time position system. The MC-lab is often used for experimental testing of DP systems and was used for this purpose in Article 2. The water tank is also used as the underwater laboratory environment in Article 1. The laboratory is mainly used by Master students and PhD-candidates at the Department of Marine Technology, but also available for external users. The towing tank and control room are depicted in Figure 2.1.



(a) Towing tank



(b) Control room

Figure 2.1: MC-lab.

2.1.2 Trondheimsfjorden

Trondheimsfjorden is a fjord located outside the city Trondheim, Norway. The location can be seen on Figure 2.2. Trondheimsfjorden was in 2016 declared the worlds first technological test facility for autonomous vessels operating below, above and at the surface of the water. In 2019 a charging stations was placed at the bottom of the fjord, to use for experimental testing of underwater drones. Images of the environment enclosing the test facility at the bottom of the fjord are used in Article 1.



Figure 2.2: Location of Trondheimsfjorden.

2.2 Equipment and Software

This section describes the most relevant equipment and software used in applications of the appended articles.

2.2.1 BlueRov2

The BlueROV 2 is a small observation-class ROV produced by **BlueRobotics**. The ROV comes with open-source electronics and software and is highly customizable for use in inspections, research, and adventuring. BlueROV2 is depicted in figure 2.3 and the main features can be seen in Table 2.1.

Table 2.1: Bluerov2 specifications

| Parameter | Value |
|------------------|--------------------------------|
| L × H × W | 457 [mm] × 254 [mm] × 575 [mm] |
| Weight in air | 11.5 [kg] |
| Weight submerged | 0 [kg] |
| Thrusters | T-200 |
| Battery | 14.8 [V], 10 [Ah] |
| Depth rating | 100 [m] |
| Camera | Raspberry Pi Camera V2.1 |
| Onboard Computer | Raspberry Pi 3B and Navio2 |



Figure 2.3: BlueROV2 underwater vehicle

2.2.2 Robotic Operating System (ROS)

ROS is an open-ended collaboration framework for writing robot software. Despite the name, ROS is not an operating system, but more a collection of tools, libraries and conventions. ROS originates from Stanford University in the mid 2000s where researchers craved software systems intended for robotics use. Today ROS is an open source project and is used by everyone between simple hobbyists to large scale industrial systems.

2.2.3 Python

All the mentioned frameworks are implemented using Python. Python is a programming language developed under and OSI-approved open source license. The fact that it is open source makes it free to use even for commercial use. There exists libraries that are free to download and specifically developed for machine learning algorithms. Relevant libraries that has been used in the mentioned frameworks are TensorFlow and PyTorch.

2.2.4 Tensorflow

TensorFlow was developed by the Google Brain group at Google for internal use in Google [27]. Google Brain is a deep learning artificial intelligence research team at Google. TensorFlow was released to the

public at November 9, 2015, and has been a large participant for IT-companies the last years. Airbnb, Uber, Spapchat, Google, Twitter and many more are all using **TensorFlow** [28], and the contributors provides extensively material on learning the framework as well as get started with creating your own machine learning applications. The framework is an open source software library used for numerical computing and machine learning.

2.2.5 PyTorch

PyTorch was based on Torch which again is based on the Lua programming language [29]. The library is an optimized tensor library for deep learning using graphics processing units (GPU) and central processing units (CPU). The library was initially released in 2016 and was primarily developed by Facebook and Uber's "Pyro" software [30].

Chapter 3

Background Theory

This chapter considers the theoretical background for the thesis. Neural networks will be presented in order to give a better insight in how they are used in existing frameworks as well as highlighting the potential such artificial networks holds. More general understanding of machine learning will also be discussed and how this is applied to domain transfer and object detection. Existing methods and solutions will be introduced throughout the chapter

3.1 Neural Networks

When discussing machine learning, deep learning or artificial intelligence, neural networks are essential. A neural network can be defined as an artificial network inspired by the natural neural networks in the human brain [31]. Artificial neural networks are designed to perform cognitive functions, e.g. problem solving and machine learning. Neural networks have successfully been implemented in games [26], handwriting recognition [32] and even explosive detection [33]. Neural networks provides a method for defining a system too complex to be defined by a simple model, e.g. image recognition and other systems influenced by uncertainty.

In order to understand neural networks it can be advantageous to know graphs. Graphs are represented with a set of points and the connection between them. A graph consist minimum of vertices and edges, and can be represented by

$$G = (V(G), E(G)), \tag{3.1}$$

where $V(G) = Vertices$ and $E(G) = Edges$ [34]. A simple graph is illustrated in Figure 3.1.

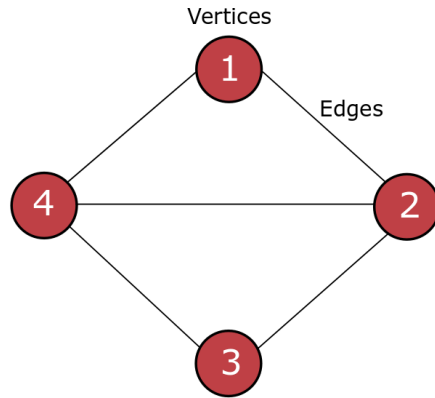


Figure 3.1: Simple illustration of a graph

The edges in a graph can either be directional or undirected. A neural network is a particular version of a graph and consists of neurons and links between the neurons. The neural network often consists of several layers of neurons and the links describe the connection between the layers. A simple neural network is depicted in Figure 3.2. Neural network terminology includes terms such as input layer, hidden layer and output layer. In a network with 3 layers like Figure 3.2, layer 1 would be the input layer, layer 2 the hidden layer and layer 3 would be the output layer. A network can contain several hidden layers and are referred to as deep networks, which will be discussed in Section 3.3.2. The term "hidden" only indicates that it is neither an input nor output layer.

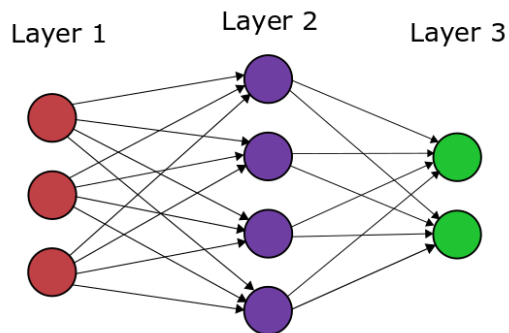
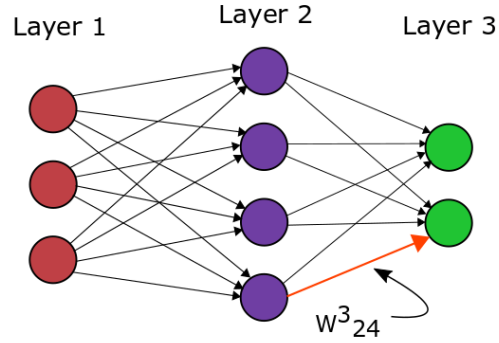


Figure 3.2: Simple Neural Network

In neural networks the links often have additional attributes. These attributes are called weights and describes the correspondence between specific neurons. Every link between all connecting neurons has individual weights, meaning the number of weights in a large neural networks can't exactly be counted on one hand. The weights can be denoted as w_{jk}^l , which represents the weight between the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer. An example is shown in Figure 3.3. The Figure illustrates the weight between neuron 4 and 2 in the 2nd and 3rd layer. The notation is taken from [35].

Figure 3.3: Weight w_{24}^3 in a neural network

During training of a neural network these weights are altered and updated. It is neither a surprise nor a secret that this requires time and computational power. To circumvent this issue, pretrained models can be utilized. Today there exist plenty of pretrained models available for research and training purposes. Pretrained models consists of already trained weights for specific problems, which can reduce training time for educational purposes. However, this assumes that the pretrained model corresponds to the problem that it is used on. A pretrained model to recognize red/green lights in traffic lights is not guaranteed to be applicable for recognizing yellow blinking lights on a subsea panel. This is because we don't really know which weights are important for the two slightly different problems, due to the complexity of the neural network.

3.1.1 Training a Neural Network

The overall goal when training a neural network is to alter weights to make the connection between neurons optimal. This will provide a network that optimally represents the desired solution to a certain problem.

Weights

Weights have been mentioned, however how they influence the output from neurons has not been explicitly explained. It was scientist Frank Rosenblatt who first introduced weights when he invented the perceptron algorithm in 1958 [36]. Rosenblatt utilized a step function to determine the output from a neuron. All the inputs to a neuron were weighted with different weights and the output, 0 or 1, was determined from whether the weighted sum was less or more than a threshold value.

$$\text{output} = \begin{cases} 0, & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1, & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases} \quad (3.2)$$

where w_j is the weight and x_j is the input. To simplify the notation of the weighted sum, a bias can be introduced to represent the threshold. If we define the bias as $b := -\text{threshold}$, it follows that

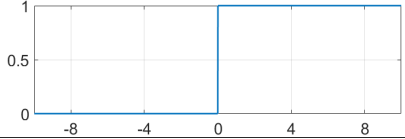
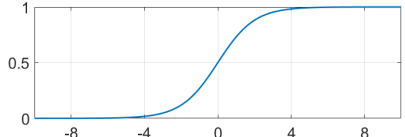
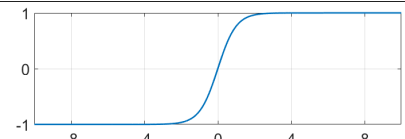
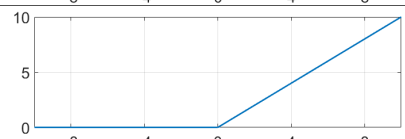
$$\text{output} = \begin{cases} 0, & \text{if } \sum_j w_j x_j + b \leq 0 \\ 1, & \text{if } \sum_j w_j x_j + b > 0 \end{cases} \quad (3.3)$$

This weighted sum is calculated for every neuron in order to determine the output. The weights then works in a way where attributes that are important to the neuron will have high weights on the incoming links and the not so important attributes will have low values.

The perceptron function involves a step function, which leads to either 0 or 1 as output. A small change in the input can therefore change the output completely and this can again have huge effects on the rest

of the network. Other functions have therefore been introduced in order to ensure a gentler relation between changes in input and output. These types of functions are often referred to as activation functions and denoted by σ . The most commonly used activation functions are the sigmoid function, the hyperbolic tangent function and the ReLu function. They take the weighted sum as input and are all illustrated in Table 3.1 along with the step function. Note that $z = \sum_j w_j x_j + b$.

Table 3.1: Commonly used activation functions

| | | |
|------------------|---|---|
| Step function | $\sigma(z) = \begin{cases} 0, & \text{if } z \leq 0 \\ 1, & \text{if } z > 0 \end{cases}$ |  |
| Sigmoid function | $\sigma(z) = \frac{1}{1+e^{-z}}$ |  |
| tanh | $\sigma(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ |  |
| ReLU | $\sigma = \max\{0, z\} = \begin{cases} 0, & \text{if } z \leq 0 \\ z, & \text{if } z > 0 \end{cases}$ |  |

Cost function

It was stated in Section 3.1 that the weights are altered during training of a network. In order to quantify how well the network behaves with certain weights and biases, a cost function is introduced. The cost function is often represented by a mean squared error (MSE) function, and is denoted

$$C(w, b) := \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2, \quad (3.4)$$

where n is the total number of training inputs, a is a vector of outputs from the network with x as input. The sum is over all the individual training inputs, x where $y(x)$ is the desired output. The aim of a training algorithm would be to minimize the cost function. In the literature the cost function is also referred to as the loss function. The desired outcome for a near-perfect network would be to get $C(w, b) \approx 0$ by altering the weights and biases. In order to solve such a minimization problem, gradient descent, which is a commonly used method for minimization problems, can be used. In order to find the minimum of the cost function using gradient descent, the gradient of $C(w, b)$ is considered

$$\nabla C(w, b) := \left(\frac{\partial C}{\partial w}, \frac{\partial C}{\partial b} \right)^T. \quad (3.5)$$

Thus the aim is to find the minimum cost function by following the gradient, $\nabla C(w, b)$. As illustrated in Figure 3.4, the red dot represents a global minimum which is the desired point.

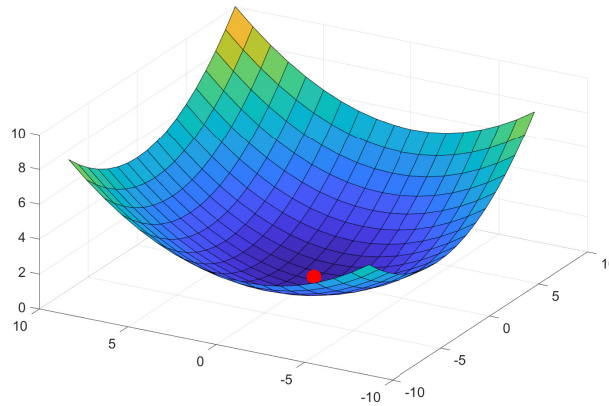


Figure 3.4: Optimal gradient

It is easy to see in this plot that the gradient, $\nabla C(w, b)$, will be purely positive and you will move towards the minimum value by moving towards decreasing gradient. In other words, changes in w_i and b_i that provides decrease in magnitude of C_i is desired. Mathematically this is represented with the gradient descent algorithm

$$(w, b)' = (w, b) - \eta \nabla C(w, b) \quad (3.6)$$

where η is referred to as the learning rate and $(w, b)'$ is the updated values of the weights and biases. When the gradient becomes 0 the function knows it is at the minimum. However, cost function tends not to look like such an optimal plot. A often occurring problem from gradient descent is the saddle point problem. The saddle point problem considers a situation where you get stuck on a saddle, believing you are at the global minimum. Such a point is illustrated in Figure 3.5 below.

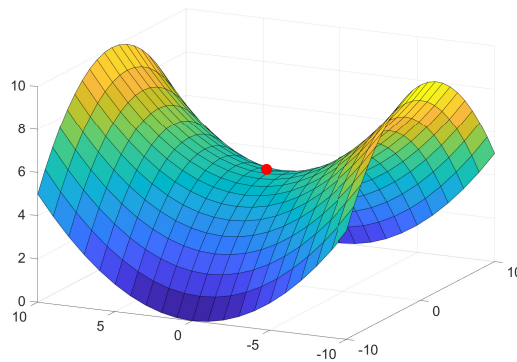


Figure 3.5: The saddle point problem

Stochastic gradient descent

To circumvent this problem the 2nd derivative can be computed. Then the algorithm can be ordered to follow a positive 2nd derivative, meaning the gradient closes into 0. This derives a new problem, the required computational time if 2nd derivatives of the cost function should be considered. Another algorithm called stochastic gradient descent has been proposed which can solve this issues. This method was probably first proposed by [37] in 1951, and suggests a method were the samples are determined randomly instead of the way they appear in the input set. This generates a noisy gradient, and for

non-convex minimization problems, considering a noisy gradient is actually not a bug, but rather a desired feature. The stochastic gradient descent equation would then be denoted

$$(w, b)' = (w, b) - \eta \nabla C_i(w, b), \quad (3.7)$$

where the difference from (3.6) is that now $C(w, b)$ is approximated by a gradient at a single sample, $C_i(w, b)$, instead of the average of the weighted sum from (3.4).

Architecture

As stated, the training of a neural network consists of minimizing the cost function with regards to weights and biases. Another important parameter concerning the overall capability of the neural network is its architecture. The architecture concerns number of layers, number of neurons in each layer, connections between neurons etc.

A nematode worm possesses a total of 302 neurons [38]. Still this presumably unintelligent worm is capable of performing complex tasks super computers today have troubles with. This is due to the complexity of the yet unknown inner mechanisms and architecture of a worm's biological neural network. As stated before, artificial neural networks are inspired from the biological networks in human brains. However, the immense complex brain is still not fully understood even by scientists who have devoted large part of their professional life to investigating the human brain. Artificial neural networks' architecture are therefore just a mere sketch of the complex biological version. Still, through the 4th industrial revolution we now experience, new methods, algorithms and frameworks emerge rapidly. A type of neural network that has proven to be successful in image recognition and classification is convolutional neural networks.

3.2 Convolutional Neural Networks (CNN)

CNN's consists of three layers

- Convolution layer
- Pooling layer
- Fully Connected layer

3.2.1 Convolution Layer

[39] defines CNN as

”Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.”

CNNs can be traced back to Fukushima and his Neocognitron [40]. He proposed a hierarchical multi-layered neural network performing robust visual pattern recognition. Adding convolution to the neural networks applies the definition

$$(f * g)(t) \equiv \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau. \quad (3.8)$$

$(f * g)(t)$ is a entirely new function based on $f(t)$ and $g(t)$. It can be seen as the weighted average of the function $f(\tau)$ at the time instant t where the weighting is given by $g(-\tau)$. In other words, convolution describes the output in terms of the input.

Relating this theory to the neural network terminology we can write

$$s(t) = (x * \omega)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)(t - \tau), \tag{3.9}$$

where the output $s(t)$ is often referred to as the feature map. The inputs x and ω is often referred to as the input and kernel, respectively. As seen by (3.9), this is the discrete version of the continuous definition of convolution from (3.8). The problems this thesis will address are with regards to images, therefore it would be more relevant to talk about multidimensional arrays. If an image, \mathbf{I} is used as the input, which is normally represented with two dimensions, the kernel \mathbf{K} should be two-dimensional as well. (3.9) can then be altered to a two dimensional version

$$\mathbf{S}(i, j) = (\mathbf{I} * \mathbf{K})(i, j) = \sum_m \sum_n \mathbf{I}(m, n)\mathbf{K}(i - m, j - n). \tag{3.10}$$

Discrete convolution can now be seen as matrix multiplication. If the networks are regarded as matrices the kernels are often smaller matrices that extract the desired information from the input. A known kernel in CNN is the edge detection kernel. This is a kernel that can be applied to an image to detect edges [41], and can be represented by the matrix

$$\mathbf{K}_{edge} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.11}$$

This kernel is multiplied with a matrix representing a picture in matrix form. This mathematical procedure is shown in Figure 3.6.

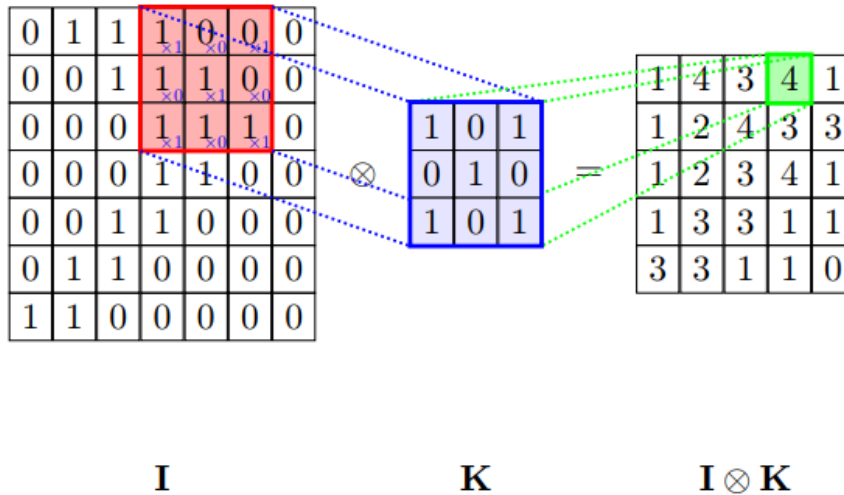


Figure 3.6: Convolution in matrix perspective [41]

A (3x3) kernel will then reduce a (n,m) matrix to (n-2,m-2), and the information in the resulting feature map will represent the desired information. However, if the input is a small image and several layers of convolution can result in an undesirable small feature map. Padding is introduced to cope with this, and involves adding additional rows and columns of 0's at the beginning and end of the

input matrix so the resulting feature map size won't shrink. This method can be illustrated with

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \xrightarrow{\text{padding}} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (3.12)$$

Figure 3.7 shows a picture where the edge detection kernel (3.11) is used. This is a black and white picture and as seen, the kernel perfectly locates the edges in the picture.

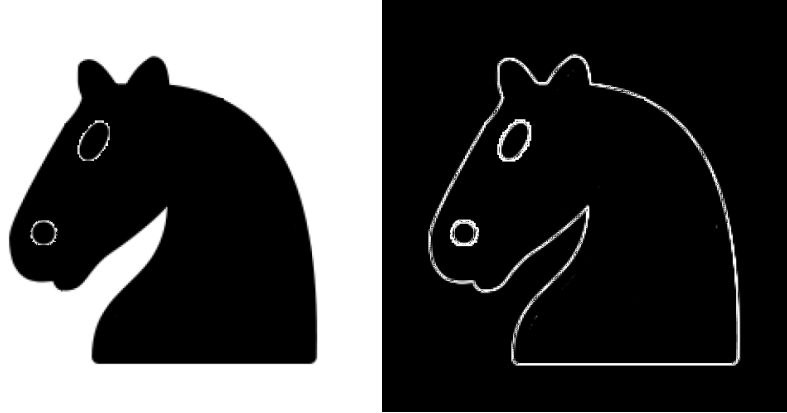


Figure 3.7: Edge detection applied to a black and white picture

Figure 3.8 illustrates another picture where the edge kernel is applied. This is a colored picture with a lot more variance than the picture of the chess piece. However, it can be seen that the kernel still locates the edges quite good. In the resulting image to the right it is possible to see the outlines of the dog even though there is a great deal of noise from the grass.



Figure 3.8: Edge detection applied to a real picture [42]

3.2.2 Pooling Layer

When the convolution operation is done, pooling can be executed in order to decrease the feature map dimensions. If pooling is executed, padding is normally not done. It would be counter intuitively to ensure maintaining the dimension of the feature map if is decreased in the very next operation. Regarding pooling, either max pooling or average pooling can be executed. The operation consist of

running the input through a filter to extract the desired information and neglect the rest. Max- and average-pooling is illustrated in Figure 3.9

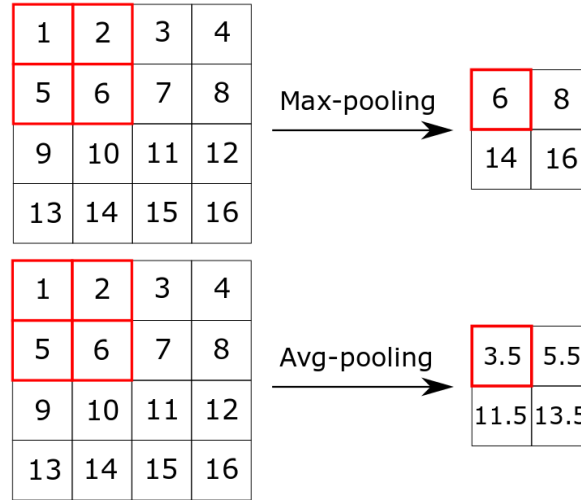


Figure 3.9: Max- and average-pooling

Pooling does not have any parameters to learn, only hyper-parameters that have to be selected. These parameters are the size of the pooling filters and the stride which is the number of elements the filter moves between operations. In the example in Figure 3.9 the size of the filter is 2×2 and the stride is 2.

3.2.3 Fully Connected Layer

Fully connected layers acts like a standard single neural network layer. Thus, it consists of weights and biases that should be trained. Note that the connected layers have sparse connections. This means that nodes are not connected to every single node in the following layer. This results in fewer connections, thus fewer operations and a decrease in computational complexity. Figure 3.10 illustrates the architecture of a CNN inspired by the LeNet-5 network including convolution layers, pooling layers and fully connected layers.

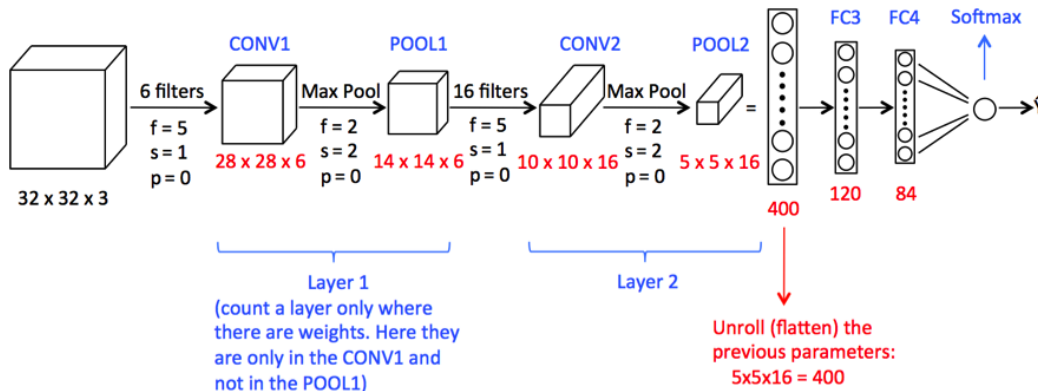


Figure 3.10: Fully connected CNN [43]

This network consists of two convolutional layers, two pooling layers and two fully connected layers. Note that each pooling layer follows directly a convolution layer. The flatten procedure organize

all parameters in a single vector which is used as input to the neural network. This is to ensure the standard shape of an input layer. Note that the layers in the convolutional part of the network consists of 3D matrices. This is due to the shape of the initial input, e.g. an image. An image consists of red-green-blue (RGB) values, thus the input has 3 layers, one for each RGB value.

3.3 Machine Learning

In the last few years, new methods and articles regarding machine learning have regularly been published. The interest in machine learning itself has experienced a substantial growth. Figure 3.11 illustrates the trend of how many times *machine learning* is searched for on the internet the last 10 years [44]. It has increased tenfold in the last 10 years.



Figure 3.11: Trends of *machine learning* the last 10 years

Machine learning applications have achieved state-of-the-art performances in multiple disciplines. Google’s *AlphaGo* has beaten the worlds best human Go player, and is arguably the strongest Go player in history [26]. *InnerEye* by Microsoft uses machine learning to develop image diagnostic tools in order to detect tumors etc. In addition to the fields of games, technology and medicine, machine learning approaches are also believed to have a dramatic impact in the fields of economics [45] in the short future. Thus, it is safe to say that machine learning will, at some extent, impact the majority of humans alive today.

Machine learning can be traced back to Arthur Samuel when he proposed a definition in 1959 [46].

”Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed”.

However this definition could be interpreted as a bit vague. Almost 40 years later in 1997, Tom Mitchell came with a more clear definition [47]

”A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E”.

Article 1 concerns the problem of reducing the reality gap. Relating Mitchell’s definition to this specific problem, the task T involves reducing the gap between a simulated environment and the real world. The performance measure P is related to at which degree transformation between the domains is achieved. The Experience E involves the information and features the frameworks manages to extract from the dataset.

3.3.1 Learning Algorithms

From Mitchell’s definition, the goal of machine learning is clear. However, how to achieve this goal is not yet stated. Different machine learning algorithms can be used to achieve a self learning computer program.

Supervised Learning

Supervised learning algorithms receives a dataset with labeled data. An example of such a dataset could be images of cats and dogs where every image has a label stating if the image contains a cat or a dog. During training the algorithm receives input and suggest an output. Afterwards the algorithm is told the "correct answer" and learns from the experience. The algorithm should later be able to recognize other cats and dogs based on the training. However, if it receives an image very different than the ones from the training set, it could experience difficulty. The idea is in general to learn the mapping $Y = f(X)$.



Figure 3.12: Supervised learning

The example with cats and dogs is a classification problem. Supervised learning can also solve regression problems. A typical regression problem is to estimate the selling price of a house by considering number of rooms, size, building year etc. The system would then learn a mapping from these parameters to the selling price by training on houses with known selling price.

Unsupervised Learning

Unsupervised learning algorithms receives an unlabeled dataset. This means the algorithms should learn without knowing the exact "correct answer". The supervised term originates from that the algorithm has an instructor which shows the algorithm what to do. An unsupervised learning algorithm has to manage without this instructor. Instead the algorithms learns useful properties and structures of the dataset.



Figure 3.13: Unsupervised learning

Unsupervised learning can be divided into clustering and association problems. Clustering concerns discovering groupings in the dataset, e.g. sorting an item by color. Association concerns finding connections/rules in the dataset, e.g. students getting straight A's also tends to be organized and to work a lot. A known unsupervised learning method is Autoencoders, which are neural networks that aims to copy their inputs to their outputs. Autoencoders will be discussed in Section 4.1.

Reinforcement Learning

Reinforcement learning works in a different way than supervised- and unsupervised learning. Instead of just interacting with a fixed dataset, a reinforcement learning algorithm interacts with the environment, which generates a feedback loop between the learning system and its experience [39].

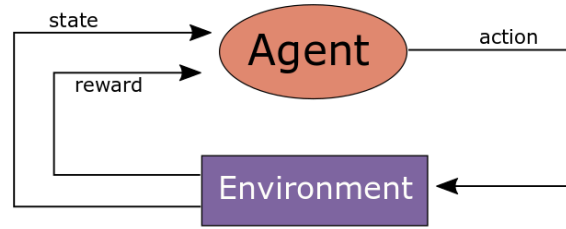


Figure 3.14: Reinforcement learning

It can be explained by figure 3.14. The learning concerns how an agent should take some action in an environment in order to maximize some reward [48]. [49] proposed a reinforcement learning algorithm called Q-learning. The goal of this algorithm is, simple put, a way of learning a policy in order to tell agents which action is optimal under different circumstances.

3.3.2 Deep Learning

Deep learning is a subset of machine learning and can be defined within any of the learning algorithms discussed above. The expression "deep" refers to the depth of the networks used in the learning. For deep learning algorithms, several hidden layers are used. The architecture of these neural networks are inspired by the biology of the human brain [50]. However, unlike the human brain where every neurons can connect together within certain areas, deep neural networks are divided into discrete layers with specific connections and data transactions. The difference from an ordinary network vs a deep network is illustrated in figure 3.15 below.

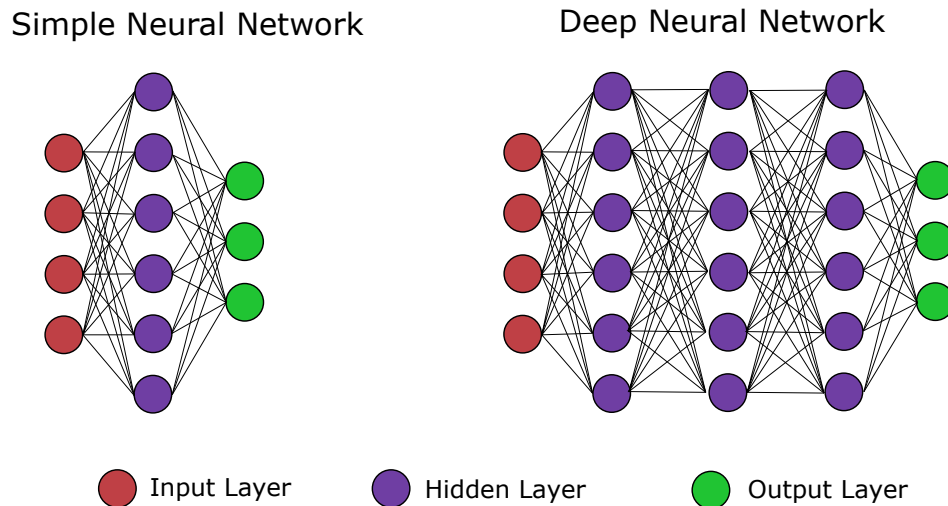


Figure 3.15: Ordinary network vs deep network

A deep learning algorithm can detect if a prediction is wrong on its own, while a simple machine learning algorithm needs an external invigilator to determine this. This is a result of the extra hidden layers in the network. [51] studied the complexity of functions computed by deep neural networks in terms of their number of linear regressions. Additional hidden layers increased the number of regressions, but also provided superior performance.

3.4 Vision Based Learning

The increasing improvement of CNN produce a solid foundation to develop vision based learning. In this section relevant learning methodologies for the articles are introduced. The relevant topics to be interpreted are domain transfer and object detection.

3.4.1 Domain Transfer

Transfer learning is a substantial, however interesting problem in machine learning. Human brains are experts at knowledge transfer. Consider an example of a chef working in a french kitchen. He/she is probably more equipped to work in a sushi restaurant than an engineer from Norway. The ability the chef possesses where he/she can transfer his knowledge to another related task is present in all humans. This might be perceived as a basic trait of the human intelligence, but is in fact extremely complicated to establish as a computational ability. A robotic arm can be trained to sort red and yellow cubes. However, the model often run into problems if the color of the cubes change to blue and green. Or if the shape changes to triangles or simply the lighting setting changes. Models trained in a simulated environment often experience the same problems when they are applied in the real world. This is referred to as the reality gap. Different approaches has been tested to reduce this gap between a simulated environment and the real world. [52] developed an object detector that trained only using simulations. They demonstrated how their object detector could achieve high enough accuracy when tested in real life after only being trained from simulations. The authors focused on a robotic arm that would grasp desired objects in a cluttered environment. The authors objective was to perceive the real environment as just another variation. This is an attempt to bridge the reality gap that is present in simulations today. As most methods within domain transfer, their methods architecture was based of CNN. The architecture consisted of five groups of convolutional layers and two fully connected layers, and pooling between each of the groupings of convolutional layers.

The main idea is to make machines manage to transfer knowledge between different domains and execute different related tasks. A related goal is to be able to execute the same tasks, however in different environments. Regarding underwater operations the latter one is particularly interesting. Training machines in an underwater environment can be extremely time consuming and error-prone due to harsh environment. However, if machines are trained exclusively in simulations, the transfer of knowledge to the real world could also generate failure. Generating robust techniques for transferring the knowledge between domains is therefore of immediate interest for operators in this market.

3.4.2 Object Detection

Vision based object detection have exploded in the last few years. With the arrival of autonomous cars, tremendous amounts of resources are put into research and development of the robustness of such systems. **Tesla** has in the last years been the quarterback pushing this development. Even though it was not revealed before 2014, every **Tesla** vehicle ever to enter the streets were built with the intention of one day become autonomous. The vehicles have in fact gathered data since the first **Tesla Roadster**, owned by **Tesla** CEO himself, Elon Musk, entered the streets in 2008. The gathered data consist of sensor information and are used by engineers at **Tesla** for everything between improve faults on the vehicles to train neural networks in autonomous appliances. This includes object detection applications.

Object detection is commonly separated into two types (See Figure 3.16). One type follows the traditional object detection procedure. In essence, this means identifying region proposals and classifying the proposals into object categories. In the literature this is known as Region Proposals Networks (RPN). Such methods consider, at some extent, the same methodologies as the human brain. The methods perform an initial scan of a scenario before separating the information into regions of specific interest. The second type follows a classifier-based approach or a regression problem. These

methodologies both arrive from the supervised learning branch of machine learning (See Figure 3.17). Thus, they share the overall objective of supervised learning to learn the mapping $f(\cdot)$ in $y = f(x)$, as discussed in Section 3.3.1. The pivotal difference between the classifier and regression approaches is the type of output. Classification approaches aim to learn the mapping to a discrete or categorical output, while regression approaches aim to learn the mapping to a continuous or numerical output.

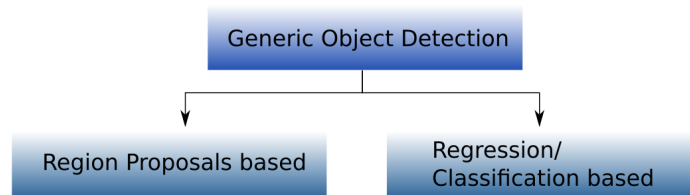


Figure 3.16: Object Detection Methodologies

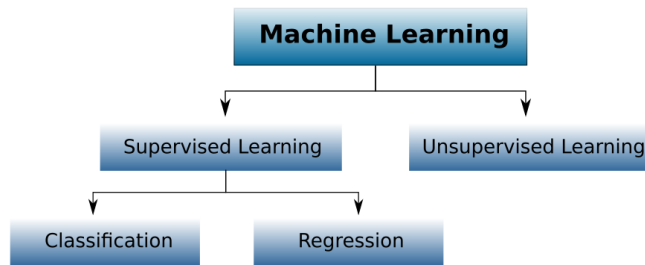


Figure 3.17: Machine Learning methodologies

Associating vision-based object detection with underwater applications is increasingly relevant. Such detection objectives can be solved with machine learning approaches such as classification [53], salient feature detection [54] or object detection [55, 56]. Object detectors build with neural networks can include a large variety of properties depending on the specific application. Vision based detectors commonly considers edges, color differences and optical flow. Such properties are frequently considered in combination with one another when building the neural networks.

Chapter 4

Methodology

This Chapter considers the contemplated methodology for the thesis. Existing state of the art frameworks are presented and necessary knowledge about the methodology considered in the articles are given.

4.1 Autoencoders

As previously stated, Autoencoders are an unsupervised learning method. Autoencoders are fundamental for the architecture of neural networks when considering optical data as the contemplated medium. The network functions by compressing the input into a latent-space representation before reconstructing the output from this latent space [57]. This mapping is illustrated in figure 4.1, where x is the input, $h = f(x)$ encodes the input to the latent space, and $r = h(z)$ decodes the latent representation to the output. Thus the whole process is described with $r = h(f(x))$ and the goal is to achieve $r \approx x$. This process is carried out in order to extract desired features between the output and input.

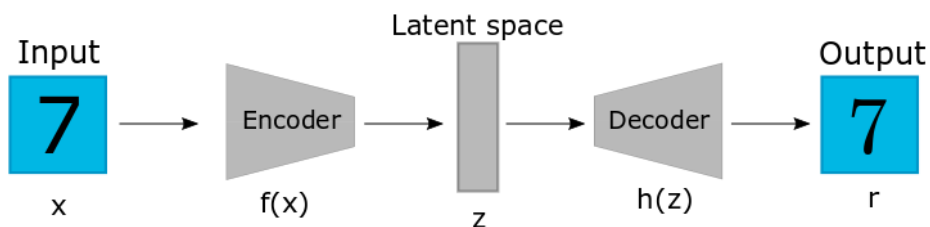


Figure 4.1: Architecture of autoencoders

An important property of the autoencoders is that the latent space, z , must have a smaller dimension than x . If they are the same size, the encoder can simply copy the entire input without learning any specific features. In this case even linear encoders and decoders can learn the mapping. By restraining the dimension of the latent space, the autoencoders must learn to extract only the most important features of the input in order to reconstruct a satisfying copy. There exists several types of autoencoders. Four regularly mentioned types are

- Vanilla autoencoder
- Multilayer autoencoder
- Convolutional autoencoder

- Regularized autoencoder

The difference in these four methods are quite self-explanatory. The vanilla autoencoder is a simple version consisting of three layers, i.e. a neural network with one input layer, one hidden layer and one output layer. The encoder and decoder are then simply the mapping between the input and the hidden layer and the mapping from the hidden layer to the output layer respectively. The multilayer autoencoder simply consists of several hidden layers. Convolutional autoencoder includes convolutional operations as explained in Section 3.2. Regularized autoencoders are somewhat separated from the other types. These autoencoder encourages the model to have other abilities than copying the output from the input. The regularized autoencoders are typically described by two types, sparse autoencoders and denoising autoencoders. The sparse autoencoders must respond to unique statistical features of the training dataset due to the sparsity. Thus the model can learn useful features of the dataset in addition to be an ordinary copy from input to output. For the denoising application, noise are added to the input and the denoising autoencoder learns to remove it. This way the autoencoder will extract the most important features and learn a more robust representation of the data.

4.2 Generative Adversarial Networks (GAN)

In the field of domain transfer, Ian Goodfellow came with a small breakthrough when he in 2014 introduced GAN. This network consist of a combination of two networks, a generator and a discriminator. The generator aims to produce content, while the discriminator determines the level of authenticity of this content. They learn simultaneously and compete against each other, in what can be described as a zero-sum game. An easy way to imagine the two networks, is to look at the generator as a criminal trying to produce fake money, and the discriminator as the police who determines if the money are real or fake. If the police rightly classifies the money as fake, the criminal (if he/she is smart) will learn from this experience and produce more realistic money. Along the way, the police will also get better at classifying the fake money and they will simultaneously get better at their respective job.

GAN is based on differentiable generator networks, which transforms samples of latent variables \mathbf{z} to samples \mathbf{x} or distributions of samples \mathbf{x} using a differentiable function $G(\mathbf{z})$ [39]. A function is said to be differentiable if the derivative of the function exists at each point in the functions domain. GAN works on the principle of comparing generator networks with discriminator networks, in other words a generator and a discriminator compete against each other. The principle is quite easy. The generator produces samples $\mathbf{x} = G(\mathbf{z})$, and the discriminator attempts to determine if the samples are produced by the generator or if they come directly from the training set. The discriminator produces a probability given by $D(\mathbf{x})$, indicating the probability that \mathbf{x} is a real sample rather than a fake sample produced by the generator. The end-goal of GAN is that the Discriminator will be unable to distinguish the real samples from the fake and produce a constant probability of 0.5.

As stated, an often used method for learning GAN is looking at a zero-sum game. Consider a function $v(G, D)$ that determines the payoff of the discriminator. In a zero-sum game the gains and losses of the participants are exactly balanced and equal to zero [58]. Thus the generator receives $-v(G, D)$ as its own payoff. During learning both participants have a desire to maximize their payoffs, until the learning reach convergence

$$G^* = \arg \min_G \max_D v(G, D). \quad (4.1)$$

An often chosen function for v in this example is

$$v(G, D) = \mathbf{E}_{\mathbf{x} \sim p_{data}} \log D(\mathbf{x}) + \mathbf{E}_{\mathbf{x} \sim p_{model}} \log(1 - D(\mathbf{x})). \quad (4.2)$$

This function concerns the expected value of $D(\mathbf{x})$ with respect to the training data and the model. The discriminator will focus on learning to correctly classify samples as real or fake, while the generator will simultaneously try to generate as real looking samples as possible to fool the discriminator. These

two participants will learn in parallel to each other. This model can be highly under-constrained, but there exist several published methods and frameworks solving this.

[59] proposed a framework called Coupled Generative Adversarial Network (CoGAN) for learning joint distributions between individual domains. The paper aimed to obtain a learning based on the joint distributions between domains rather than learning from corresponding images in different domains. This simplifies the requirements of the datasets, because with CoGAN they didn't require corresponding images in the different domains. The framework discovered the joint distribution instead. They applied the CoGAN framework for color and depth images, and also on face images with different attributes and demonstrated successfully image transformations between domains.

[60] illustrates methods for unsupervised image-to-image translation. They do this by learning a joint distribution between individual domains. Their framework assumes there exists a shared-latent space and is based on the CoGAN framework from [59]. The shared-latent space assumptions assumes that a pair of corresponding images in different domains can be mapped in the same latent domain. A latent space is a simpler representation of the images [61]. A shared-latent space between two domains could be a way to represent the desired features in different images. By transferring the images to the shared-latent space these shared features could be simpler to extract. By assuming there exist such a shared-latent space [60] demonstrated image-to-image translation between two domains without any corresponding images in the training datasets. They also discovered two limitations of the framework. The translation model was unimodal due to the Gaussian latent space assumption. A unimodal model means there exist only one peak, i.e. one right answer. The second limitation was possible unstable training due to the saddle point searching problem. The saddle point searching problem is a well known issue within regression analysis and machine learning frameworks. The issues consist of difficulties in distinguishing saddle points from maximums and minimums. This problem has been discussed in Section 3.1.1. Another framework deriving from GAN is CycleGAN which implements a cycle consistency loss in addition to the already implemented adversarial loss.

4.2.1 CycleGAN

CycleGAN is a method to perform image-to-image translation between domains without paired images in each domain [62]. The authors have created methods using both TensorFlow and PyTorch. The image-to-image translation are achieved without paired images by adding an additional mapping function F in addition to G . Figure 4.2 represents the architecture of this translation.

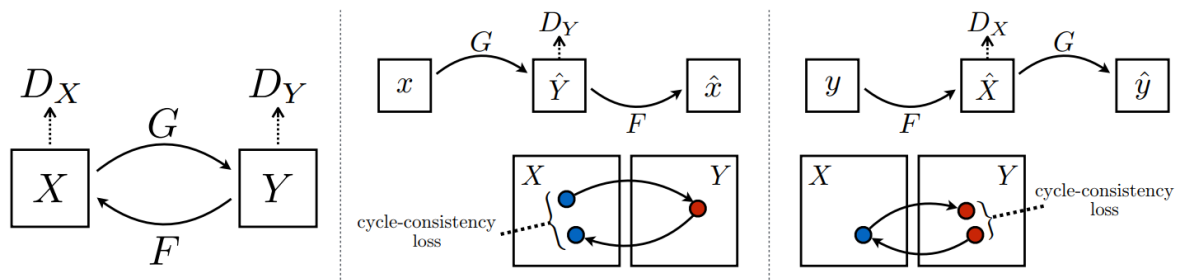


Figure 4.2: Architecture of the CycleGAN method taken from [62]

From the architecture of CycleGAN it can be seen that when adding the additional mapping function $F(y)$, an additional discriminator D_Y is also necessary. D_Y attempts to distinguish images from domain x and generated images $F(y)$, and similarly D_X tries to distinguish between images from y and $G(x)$. The objective function now becomes twofold and includes the adversarial loss and a new cycle consistency loss. The adversarial loss can be defined as the payoff function in the zero sum game

explained in Section 4.2 and (4.2). This gives an adversarial loss function represented by

$$L_{GAN}(G, D_Y, X, Y) = \mathbf{E}_{y \sim p_{data}} \log D_Y(y) + \mathbf{E}_{x \sim p_{data}} \log (1 - D_Y(G(x))). \quad (4.3)$$

The cycle consistency loss is illustrated in the middle and right figures in Figure 4.2. Implementing a desired cycle consistent mapping means that for each image, x or y , you want the original image reconstructed after the image translation cycle, i.e., $X \approx F(G(X))$ and $Y \approx G(F(Y))$. The cycle consistency loss can thus be defined with

$$L_{cyc}(G, F) = \mathbf{E}_{x \sim p_{data}} \|F(G(x)) - x\|_1 + \mathbf{E}_{y \sim p_{data}} \|G(F(y)) - y\|_1. \quad (4.4)$$

The objective then becomes a sum of the adversarial loss and the cycle consistency loss, and can be represented with

$$L_{CycleGAN}(G, F, D_x, D_y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F), \quad (4.5)$$

where λ determines the relative importance of the two objectives. Notice that there are used two functions for adversarial loss. This is in order to ensure the losses for mapping between both domains are accounted for. Remember that a loss function is equivalent to the cost function discussed in Section 3.1.1. Motivated by (4.1), the objected becomes to solve

$$G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} L_{CycleGAN}(G, F, D_x, D_y). \quad (4.6)$$

This model can be viewed as training two autoencoders, one mapping the $F(G(x)) \approx x$ and one mapping $G(F(y)) \approx y$. It should be noted that this method have special internal structure and therefore not be seen in direct comparison with the autoencoders described in Section 4.1. Both the generators and discriminators use ReLu functions as the activation functions. The ReLu function in the discriminator is a leaky ReLu with slope 0.2. A leaky ReLu function is similar to the function described in Section 3.1.1, but now the function is altered to $\sigma = \max\{0.2z, z\}$. Preliminary results from CycleGAN are depicted in Figure 4.3.

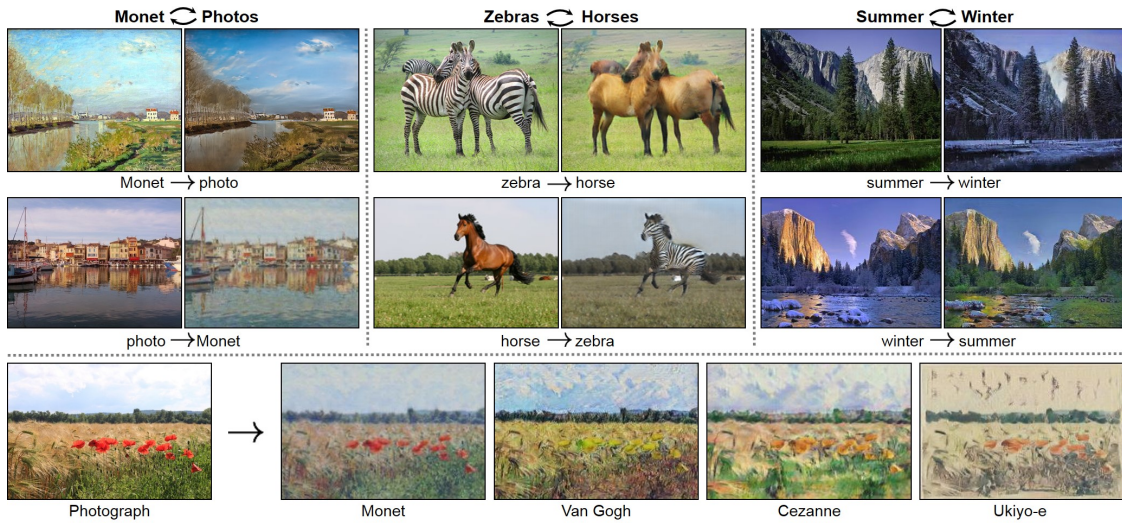


Figure 4.3: Published results from [62].

4.3 Object Detection

In object detection schemes, real-time pose of the object is generally of interest, thus stereo cameras are often used [63, 64, 65]. Stereo cameras employ two or more lenses in order to simulate humans

binocular vision, thus capturing spatial features in images to get a 3D representation. Moreover, in machine learning applications where labeled data is needed, 3D images could be complicated to process and labeling of such images are difficult. Processing standard monocular camera images and labeling ordinary RGB 2D images is much less complicated. However, this requires alternative methods for extracting spatial features. Underwater object detection using monocular camera has previously been investigated [66]. The authors used visual features such as color and intensity as well as investigating light transmission information. However, spatial features were not considered. [67] presented a 3D object detection method based on 2D images. They modified the known You Only Look Once (YOLO) algorithm [68] to receive input images from different angles and reconstruct a three-dimensional representation with a 3D bounding box locating the object. The aforementioned YOLO algorithm is one of the recent most successful real-time object detection algorithms. In contrast to several other object detection methods, YOLO identifies the detection problem as a regression problem as apposed to a classifier-based approach and the algorithm consist of a single neural network. These specifications provides a framework able to process images at real-time at 45 frames per second. In addition, the algorithm generalizes well to different domains, which makes it ideal for new applications.

A well known classifier object detector is the R-CNN [69]. The name derives from the combination of regional proposals and convolutional neural networks. The R-CNN algorithm proposes 2000 regions per image and works with these regions in an attempt to classify them in order to locate the object. 2000 regions per image is a lot to process and is computational expensive. The algorithm uses about 47 seconds per image, which makes it irrelevant regarding real-time detection. A modified version, Fast R-CNN, proposed a slightly different approach. Instead of proposing 2000 regions per image, the modified version feeds the input image to a CNN and outputs a convolutional feature map. The proposed regions from the feature map is fed through a regions of interest (RoI) pooling layer and a fully connected layer. Then a softmax layer is used in order to predict the class of the object as well as the regions for the bounding boxes. As for R-CNN, Fast R-CNN uses selective search which is a slow and time-consuming process. Faster R-CNN, a further modification, circumvents the selective search algorithm. The algorithm uses a separate network instead of selective search in order to predict the region proposals. This results in a much faster network. Faster R-CNN is the only network out of these three that is fast enough to be relevant for real-time object detection.

The mentioned R-CNN networks are all Region Proposal Networks (RPNs). [70] proposed a novel proposal generation method called Enhanced Region Proposal Network (ERPNN) inspired by the RPN in Faster R-CNN. In ERPNN four improvements were presented compared to Faster R-CNN.

- A deconvolutional feature pyramid network (DFPN) is introduced in order to improve the region proposals.
- Novel anchor boxes were designed with interspersed scales and adaptive aspect ratios. This increased the capability of the object localization.
- A particle swarm optimization (PSO) based support vector machine (SVM) was developed in order to distinguish the positive and negative anchor boxes.
- Improvement of the classification part of multi-task loss function in RPN, which resulted in strengthened effect of classification loss.

ERPNN surpassed Faster R-CNN in both speed and accuracy when tested on the PASCAL VOC 2007, 2012 [71] and MS COCO datasets [72].

Single Shot Detector (SSD) uses a classification/regression based approach and, hence, does not require object proposals, and encapsulates all computation in a single network [73]. Consequently the algorithm is fast and suitable as a real-time object detector. SSD also proved to be more accurate than the YOLO versions available at the time. However, new versions of YOLO has emerged since then. The version YOLOv3 has proven to be just as accurate as SSD, however three times faster [74]. Still YOLOv3 is not as fast as it's predecessor, where YOLOv2 could run on a Titan X at 45 FPS while YOLOv3 is limited to about 30 FPS. This is due to the increased complexity of the underlying network called Darknet. YOLOv2 utilize Darknet-19 consisting of a total of 30 layers, contained origi-

nally a 19-layer network with additional 11 layers for object detection. YOLOv3's network Darknet-53 contains 53 layers trained on ImageNet with 53 more layers for object detection. This gives a network with 106 layers. This increased complexity is the cause of decreased speed as well as increased accuracy. Darknet-53 incorporated some new important elements which Darknet-19 did not contain, such as residual blocks, skip connections and upsampling. See Table 4.1 for the Darknet-53 layers. Another issue YOLOv3 has addressed is the fact that previous versions have struggled with detecting small objects. In YOLOv3, predictions are made at three different scales, and after each detection, layers are upsampled. The upsampling helps the network learn fine-grained features, which are advantageous for detecting small objects. The predictions are conducted with a convolutional layer which uses 1x1 convolutions. For each scale 3 boxes are predicted. Each prediction consist of 3 bounding boxes, an objectness score and class scores. The predictions consist of a convolutional layer with size $1 \times 1 \times (3 * (1 + 4 + \#C))$, where $\#C$ is the number of classes the detector considers.

Table 4.1: Darknet-53

| | Type | Filters | Size | Output |
|----|---------------|---------|-----------|-----------|
| | Convolutional | 32 | 3 x 3 | 256 x 256 |
| | Convolutional | 64 | 3 x 3 / 2 | 128 x 128 |
| 1x | Convolutional | 32 | 1 x 1 | |
| | Convolutional | 64 | 1 x 1 | |
| | Residual | | | 128 x 128 |
| | Convolutional | 128 | 3 x 3 / 2 | 64 x 64 |
| 2x | Convolutional | 64 | 1 x 1 | |
| | Convolutional | 128 | 3 x 3 | |
| | Residual | | | 64 x 64 |
| | Convolutional | 256 | 3 x 3 / 2 | 32 x 32 |
| 8x | Convolutional | 128 | 1 x 1 | |
| | Convolutional | 256 | 3 x 3 | |
| | Residual | | | 32 x 32 |
| | Convolutional | 512 | 3 x 3 / 2 | 16 x 16 |
| 8x | Convolutional | 256 | 1 x 1 | |
| | Convolutional | 512 | 3 x 3 | |
| | Residual | | | 16 x 16 |
| | Convolutional | 1024 | 3 x 3 / 2 | 8 x 8 |
| 4x | Convolutional | 512 | 1 x 1 | |
| | Convolutional | 1024 | 3 x 3 | |
| | Residual | | | 8 x 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Chapter 5

Conclusions and Further Work

The presented work focuses on the use of visual aided machine learning applications in the underwater segment. The appended articles investigate different aspects of the development within underwater operations. Transfer learning and object detection have been investigated and how the challenging underwater environment affects different existing methods. The main conclusions of the articles are summarized as follows:

Article 1 investigated methods for reducing the reality gap, by generating a mapping between a simulated and a real environment. Simulations were conducted for two different datasets, localized in a laboratory pool and the bottom of a fjord. The datasets had a clear distinction in the level of details present in both the simulated and real environment. The dataset from the fjord included in general a higher level of details. Results suggested that the contemplated transfer learning framework, **CycleGAN**, was able to generate a mapping between the domains. The distinction in details in the two datasets also suggested that the higher level of details improved the results significantly. Further work includes keeping developing the obtained datasets and increasing the level of details in the simulated models.

Article 2: presents a dynamic positioning procedure relative to an object of interest using a small-class fully actuated underwater vehicle. The object is detected using a monocular camera and the detector is based on the deep learning detector **YOLOv3**. The detector is trained on a labeled image dataset of the object of interest. The dataset is large and the paper also presents a powerful labeling procedure. Experimental testing results prove the effectiveness of the proposed methods, where a small underwater vehicle performs DP relative to the object with small errors. Further work involves adapting the proposed methods to an underwater vehicle-manipulator system for simultaneous DP of the vehicle and gripping with the manipulator arm.

Bibliography

- [1] ResearchAndMarkets.com. *Underwater Robotics Market Size, Share Trends Analysis Report By Type (ROV, AUV), By Application (Commercial Exploration, Defense Security, Scientific Research), By Region, And Segment Forecasts, 2018 - 2025*. ResearchAndMarkets.com, 2018.
- [2] finbox.io. *Revenue CAGR (5y) for Apple Inc.* Nov. 2018. URL: https://finbox.io/AAPL/explorer/revenue_cagr_5y.
- [3] sintef.no. *Ocean Space Centre får grønt lys fra kvalitetssikrer*. Oct. 2018. URL: <http://www.oceanspacecentre.no/wp-content/uploads/2018/04/Supplerende-analyse-Ocean-Space-Centre-forkortet.pdf>.
- [4] Francisco Bonin-Font et al. “Visual sensing for autonomous underwater exploration and intervention tasks”. In: *Ocean Engineering* 93 (2015), pp. 25–44. ISSN: 0029-8018.
- [5] Qiao Xi, Thomas Rauschenbach, and Li Daoliang. “Review of Underwater Machine Vision Technology and Its Applications”. In: *Marine Technology Society Journal* 51.1 (2017), pp. 75–97. DOI: [10.4031/MTSJ.51.1.8](https://doi.org/10.4031/MTSJ.51.1.8). URL: <https://www.ingentaconnect.com/content/mts/mtsj/2017/00000051/00000001/art00009%20https://doi.org/10.4031/MTSJ.51.1.8>.
- [6] Zhe Chen et al. “Underwater salient object detection by combining 2D and 3D visual features”. In: *Neurocomputing* (2019). ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2018.10.089>. URL: <http://www.sciencedirect.com/science/article/pii/S0925231219304230>.
- [7] William J. Atto and Ronald J. Pestritto, eds. *American Progressivism: A Reader*. Lexington Books, 2008.
- [8] David Streitfeld. *Amazon Hits \$1,000,000,000,000 in Value, Following Apple*. Sept. 2018. URL: <https://www.nytimes.com/2018/09/04/technology/amazon-stock-price-1-trillion-value.html>.
- [9] Bernard Marr. *What Everyone Must Know About Industry 4.0*. June 2016. URL: <https://www.forbes.com/sites/bernardmarr/2016/06/20/what-everyone-must-know-about-industry-4-0/#2941b856795f>.
- [10] Francesco Scibilia. *Lecture notes in TTK23: Introduction to Autonomous Robotics Systems for Industry 4.0*. Oct. 2018.
- [11] Anastasios Lekkas and Francesco Scibilia. *Lecture notes in TTK23: Introduction to Autonomous Robotics Systems for Industry 4.0*. Oct. 2018.
- [12] Hope Reese. *Autonomous driving levels 0 to 5: Understanding the differences*. Jan. 2016.
- [13] Ryan W. Proud, Jeremy J. Hart, and Richard B. Mrozinski. *Methods for Determining the Level of Autonomy to Design into a Human Spaceflight Vehicle: A Function Specific Approach*. Sept. 2003.
- [14] Carlos E. Perez. *Automating High-Level Economic Thinking using Deep Learning*. Dec. 2017. URL: <https://medium.com/intuitionmachine/economic-modeling-and-deep-learning-dcd61b351cad>.
- [15] National Research Council. *Autonomous Vehicles in Support of Naval Operations*. Washington, DC: The National Academies Press, 2005. ISBN: 978-0-309-09676-8. DOI: [10.17226/11379](https://doi.org/10.17226/11379). URL: <https://www.nap.edu/catalog/11379/autonomous-vehicles-in-support-of-naval-operations>.
- [16] Ingrid Utne, Asgeir Sørensen, and Ingrid Schjøberg. “Risk Management of Autonomous Marine Systems and Operations”. In: June 2017, V03BT02A020. DOI: [10.1115/OMAE2017-61645](https://doi.org/10.1115/OMAE2017-61645).

-
- [17] *Autonomous ship project, key facts about YARA Birkeland*. 2018. URL: <https://www.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/4B8113B707A50A4FC125811D00407045?OpenDocument>.
- [18] *New DNV GL autonomous guidelines to be trialled on Yara Birkeland*. Sept. 2018. URL: https://www.marinemec.com/news/view,new-dnv-gl-autonomous-guidelines-to-be-trialled-on-iyara-birkelandi_54099.htm.
- [19] Siri Krohn-Fagervoll. *Dette skjer før Yara Birkeland blir et selvgående containerskip fra Herøya*. Oct. 2018. URL: <https://www.heroya-industripark.no/aktuelt/dette-skjer-foer-yara-birkeland-blir-et-selvgaaende-containerskip-fra-heroeya>.
- [20] seppo.net. *Sense of smell*. [Online; accessed November 17, 2018]. 2018. URL: <http://www.seppo.net/cartoons/displayimage.php?pid=1131>.
- [21] www.xyleme.com. *Want To Know What Personalized Learning Looks Like? It's Right in Front of You!* [Online; accessed November 17, 2018]. 2018. URL: <https://www.xyleme.com/want-to-know-what-personalized-learning-looks-like-its-right-in-front-of-you/>.
- [22] Ronn Torossian. *Public Relations Planning*. [Online; accessed November 17, 2018]. 2016. URL: <https://www.business2community.com/public-relations/public-relations-planning-01717601>.
- [23] brucedwatson.wordpress.com/. *Improving Vocational Education and Training*. [Online; accessed November 17, 2018]. 2018. URL: <https://brucedwatson.wordpress.com/2015/03/21/10-symptoms-of-competency-based-training/>.
- [24] Kyle Duncan. *Oklahoma entrepreneur re-launches debt collection career*. [Online; accessed November 17, 2018]. 2011. URL: <http://www.microbilt.com/news/article/oklahoma-entrepreneur-re-launches-debt-collection-career>.
- [25] Nicolas Heess, Josh Merel, and Ziyu Wang. *Producing flexible behaviours in simulated environments*. July 2017. URL: <https://deepmind.com/blog/producing-flexible-behaviours-simulated-environments/>.
- [26] *AlphaGo*. Nov. 2018. URL: <https://deepmind.com/research/alphago/>.
- [27] Wikipedia contributors. *TensorFlow*. Nov. 2018. URL: <https://en.wikipedia.org/wiki/TensorFlow>.
- [28] *An open source machine learning framework for everyone*. Nov. 2018. URL: <https://www.tensorflow.org/>.
- [29] Kirill Dubovikov. “PyTorch vs TensorFlow — spotting the difference”. In: *Towards Data Science* abs/1703.06907 (2017). arXiv: [1703.06907](https://arxiv.org/abs/1703.06907). URL: <https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-difference-25c75777377b>.
- [30] Brian McMahan and Delip Rao. *Natural Language Processing (NLP) with PyTorch*. 2018. URL: <http://dl4nlp.info/en/latest/> (visited on 11/25/2018).
- [31] *Neral Network*. Nov. 2018. URL: <https://academic.eb.com/levels/collegiate/article/neural-network/126495>.
- [32] Rolla Almodfer et al. “Multi-column Deep Neural Network for Offline Arabic Handwriting Recognition”. In: *Artificial Neural Networks and Machine Learning – ICANN 2017*. Ed. by Alessandra Lintas et al. Cham: Springer International Publishing, 2017, pp. 260–267. ISBN: 978-3-319-68612-7.
- [33] Haidong Wang et al. “Study of artificial neural network on explosive detection with PFTNA method”. In: *IEEE Nuclear Science Symposium Conference Record, 2005*. Vol. 1. Oct. 2005, pp. 471–473. DOI: [10.1109/NSSMIC.2005.1596295](https://doi.org/10.1109/NSSMIC.2005.1596295).
- [34] Jeevith Hedge. *Lecture 8 - Part 2. Fundamentals of Artificial Intelligence. Introduction to Bayesian Belief Networks*. Oct. 2018.
- [35] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.
- [36] Frank Rosenblatt. “The Perceptron - a perceivng and recognizing automaton”. In: *Cornell Aeronautical Laboratory* (Jan. 1957).
- [37] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *Ann. Math. Statist.* 22.3 (Sept. 1951), pp. 400–407. DOI: [10.1214/aoms/1177729586](https://doi.org/10.1214/aoms/1177729586). URL: <https://doi.org/10.1214/aoms/1177729586>.
-

-
- [38] J G White et al. “The Structure of the Nervous System of the Nematode *Caenorhabditis elegans*”. In: *Philosophical transactions of the Royal Society of London* 314.1165 (1986), pp. 1–340. ISSN: 0962-8436.
- [39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [40] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202. ISSN: 1432-0770. DOI: [10.1007/BF00344251](https://doi.org/10.1007/BF00344251). URL: <https://doi.org/10.1007/BF00344251>.
- [41] Mikkel Cornelius Nielsen. *Lecture notes in Underwater Robotics for Safe and Autonomous Subsea Operations*. Oct. 2018.
- [42] pets4home. *Juvenile Pyoderma In The Weimaraner Dog Breed*. 2018. URL: <https://www.pets4homes.co.uk/pet-advice/juvenile-pyoderma-in-the-weimaraner-dog-breed.html>.
- [43] Mike Cavaioni. *DeepLearning series: Convolutional Neural Networks*. 2018. URL: <https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524> (visited on 12/06/2018).
- [44] *Google trends*. 2019. URL: <https://trends.google.com/trends/explore?date=2009-07-30%202019-07-30&q=machine%20learning>.
- [45] Susan Athey. *The Impact of Machine Learning on Economics*. Jan. 2018. URL: <https://www.nber.org/chapters/c14009.pdf>.
- [46] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (July 1959), pp. 210–229. ISSN: 0018-8646. DOI: [10.1147/rd.33.0210](https://doi.org/10.1147/rd.33.0210).
- [47] Tom M Mitchell. *Machine learning*. eng. New York, 1997.
- [48] A.I. Wiki. *A Beginner’s Guide to Deep Reinforcement Learning*. 2018. URL: <https://skymind.ai/wiki/deep-reinforcement-learning#define> (visited on 10/17/2018).
- [49] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. May 1989. URL: <https://academic.elsevier.com/levels/collegiate/article/neural-network/126495>.
- [50] Michael Copeland. *What’s the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?* 2016. URL: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>.
- [51] Guido F Montufar et al. “On the Number of Linear Regions of Deep Neural Networks”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2924–2932. URL: <http://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf>.
- [52] J. Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 23–30. DOI: [10.1109/IROS.2017.8202133](https://doi.org/10.1109/IROS.2017.8202133).
- [53] Madjidi, Hossein and Negahdaripour, Shahriar. “On robustness and localization accuracy of optical flow computation for underwater color images”. In: *Computer Vision and Image Understanding* 104.1 (2006), pp. 61–76. ISSN: 1077-3142. DOI: <https://doi.org/10.1016/j.cviu.2006.07.003>. URL: <http://www.sciencedirect.com/science/article/pii/S107731420600083X>.
- [54] Zhe Chen et al. “Underwater salient object detection by combining 2D and 3D visual features”. In: *Neurocomputing* (2019). ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2018.10.089>. URL: <http://www.sciencedirect.com/science/article/pii/S0925231219304230>.
- [55] H. Qin et al. “When underwater imagery analysis meets deep learning: A solution at the age of big visual data”. In: *OCEANS 2015 - MTS/IEEE Washington*, pp. 1–5. DOI: [10.23919/OCEANS.2015.7404463](https://doi.org/10.23919/OCEANS.2015.7404463).
- [56] Md Moniruzzaman et al. “Deep Learning on Underwater Marine Object Detection: A Survey”. In: *Advanced Concepts for Intelligent Vision Systems*. Ed. by Jacques Blanc-Talon et al. Springer International Publishing, pp. 150–160. ISBN: 978-3-319-70353-4.
- [57] Nathan Hubenes. *Deep Inside: Autoencoders*. 2018. URL: <https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f> (visited on 12/07/2018).
- [58] *Zero-sum game*. 2018. URL: https://en.wikipedia.org/wiki/Zero-sum_game.
-

-
- [59] Ming-Yu Liu and Oncel Tuzel. “Coupled Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 469–477. URL: <http://papers.nips.cc/paper/6544-coupled-generative-adversarial-networks.pdf>.
- [60] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. “Unsupervised Image-to-Image Translation Networks”. In: *CoRR* abs/1703.00848 (2017). arXiv: [1703.00848](https://arxiv.org/abs/1703.00848). URL: <http://arxiv.org/abs/1703.00848>.
- [61] Julien Despois. “Latent space visualization”. In: *ai-odyssey* abs/1703.00848 (2018). URL: <https://ai-odyssey.com/2017/02/24/latent-space-visualization%E2%80%8A/>.
- [62] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.
- [63] P. J. Sanz et al. “TRIDENT An European project targeted to increase the autonomy levels for underwater intervention missions”. In: *2013 OCEANS - San Diego*. Sept. 2013, pp. 1–10.
- [64] E. Simetti et al. “Autonomous Underwater Intervention: Experimental Results of the MARIS Project”. In: *IEEE Journal of Oceanic Engineering* 43.3 (July 2018), pp. 620–639. ISSN: 0364-9059.
- [65] Jiaqi Xu and Hwan-Sik Yoon. “Vision-based estimation of excavator manipulator pose for automated grading control”. In: *Automation in Construction* 98 (2019), pp. 122–131. ISSN: 0926-5805.
- [66] Zhe Chen et al. “Monocular Vision-Based Underwater Object Detection”. In: *Sensors*. 2017.
- [67] Xia Zhao, Haihang Jia, and Yingting Ni. “A novel three-dimensional object detection with the modified You Only Look Once method”. In: *International Journal of Advanced Robotic Systems* 15 (Mar. 2018), p. 172988141876550.
- [68] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.
- [69] Ross B. Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CoRR* abs/1311.2524 (2013). arXiv: [1311.2524](https://arxiv.org/abs/1311.2524).
- [70] Yu Peng Chen, Ying Li, and Gang Wang. “An Enhanced Region Proposal Network for object detection using deep learning method”. In: *PLOS ONE* 13.9 (Sept. 2018), pp. 1–26.
- [71] Shih-Wei Lin et al. “Particle swarm optimization for parameter determination and feature selection of support vector machines”. In: *Expert Systems with Applications* 35.4 (2008), pp. 1817–1824. ISSN: 0957-4174.
- [72] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014). arXiv: [1405.0312](https://arxiv.org/abs/1405.0312).
- [73] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *CoRR* abs/1512.02325 (2015). arXiv: [1512.02325](https://arxiv.org/abs/1512.02325).
- [74] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *CoRR* abs/1804.02767 (2018). arXiv: [1804.02767](https://arxiv.org/abs/1804.02767).

Part II

Collection of Articles

Article 1

Transfer Learning in Underwater Operations

Martin Skaldebø, Albert Sans Muntadas, Ingrid Schjølberg
Submitted to the *OCEANS 2019 Marseille Conference & Exhibition* Marseille, France.

Transfer Learning in Underwater Operations

Martin Skaldebø
Dept. of Marine Technology
NTNU
Trondheim, Norway
martin.b.skaldebo@ntnu.no

Albert Sans Muntadas
Dept. of Marine Technology
NTNU
Trondheim, Norway
albert.sans@ntnu.no

Ingrid Schjølberg
Dept. of Marine Technology
NTNU
Trondheim, Norway
ingrid.schjolberg@ntnu.no

Abstract—This paper investigates a method for reducing the reality gap that occurs when applying simulated data in training for vision-based operations in a subsea environment. The distinction in knowledge in the simulated and real domains is denoted *the reality gap*. The objective of the presented work is to adapt and test a method for transferring knowledge obtained in a simulated environment into the real environment. The main method in focus is the machine learning framework CycleGAN, mapping desired features in order to recreate environments. The overall goal is to enable a framework trained in a simulated environment to recognize the desired features when applied in the real world. The performance of the learning transfer is measured by the ability to recreate the different environments from new test data. The obtained results demonstrates that the CycleGAN framework is able to map features characteristic for an underwater environment presented with the unlabeled datasets. Evaluation metrics, such as Average precision (AP) or FCN-score can be used to further evaluate the results. Moreover, this requires labeled data, which provides additional development of the current datasets.

Index Terms—Underwater robotics, transfer learning, autonomy, CycleGAN

I. INTRODUCTION

Today, underwater operations experience a shift towards use of more autonomous systems, where machine learning is believed to play a central role. Especially, regarding the ability to transfer knowledge between operator and system. Human brains are experts at knowledge transfer. This might be perceived as a basic trait of the human intelligence, but is in fact extremely complicated to establish as a computational ability. The main idea is to enable machines to transfer knowledge between different domains and execute different related tasks. An overall goal is to be able to train in a simulated domain and then execute the same tasks in the real world. Regarding underwater operations the latter one is of particular interest as the deep sea is less accessible, operations are costly and challenging.

Training machines in an underwater environment is extremely time consuming and error-prone due to the harsh environment. Moreover, if machines are trained exclusively in simulations the transfer of knowledge to the real world could also generate failure. This is referred to as the reality gap [1]. Generating robust techniques for transferring the knowledge between domains is therefore of immediate interest for operators in this market. There exist several published methods dealing with this problem, however only for specific

domains. This paper will investigate one such method for the use in the underwater domain. One of the most promising frameworks, CycleGAN, will be tested on two different datasets considering underwater environments. The datasets include real and rendered vision based pictures of subsea panels.

A. Motivation

The underwater robotic market size is claimed to reach USD 6.74 Billion by 2025 [2]. This corresponds to a Compound Annual Growth Rate (CAGR) of 13.5%. By comparison, Apple Inc.'s 5 year CAGR is, per April 2019, 9.2% [3]. The same report predicts that autonomous underwater vehicles (AUV) will account for USD 1.48 billion by 2025. The Norwegian Government is investing in the ocean space when designing the concept *Ocean Space Center*. The concept has a planned investment of 4.7 billion NOK [4].

Activity, interest and economic growth within the ocean space is in other words unquestionable, and with growth advancement in the technology is forthcoming. In the last years, machine learning has experienced a substantial growth in both media coverage and technological applications. One specific area is within vision-based navigation for autonomous systems. The wide interest and willingness to achieve progress that is shown today generates motivation for further investments in the field. Machine learning is believed to play a significant role in the shift towards autonomy.

B. Background

Underwater operations today are highly dependant on human operators. Operations previously executed by human divers are now mostly transferred to remotely operated vehicles (ROVs). Moreover, the industry is today experiencing a new shift towards more autonomous operations where ROVs becomes more independent of human operators. Increasing the level of autonomy and optimize the human-robot interaction in these operations can potentially reduce costs and increase safety [5]. A higher level of autonomy leads to new requirements and increasing the autonomous complexity. Moreover, autonomous underwater vehicles (AUVs) require higher level of autonomy than ROVs. Since global navigation satellite system (GNSS) measurements are not applicable underwater, vehicle operation in this domain lacks localization measurements and are prone to accumulation of error. Today,

the most common measurements and signal data arrives from acoustic sensors. Such signals are prone to data loss due to transmission losses, acoustic noise in thrusters, signal reflections on different surfaces, absorption loss and more. Feature extraction using camera vision are rarely used, but improvements within artificial neural networks (ANN), especially convolutional neural networks (CNN), shows promising results. In the presented work, systems using visual aided navigation will be investigated. This is mostly motivated by the rapid advance withing CNN and other computer vision frameworks building on CNN.

Although it is in the last few years CNN has been given recognition for its good results, it can be traced back to 1980 and Neocognitron [6]. He proposed a hierarchical multilayered neural network performing robust visual pattern recognition. Such networks can be defined as

”Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.” [7]

A neural network can be defined as a computer program that is inspired by the natural neural networks in the human brain [8]. Such artificial neural networks are designed to perform cognitive functions as problem solving and machine learning. Neural networks have successfully been implemented in games [9], handwriting recognition [10] and even explosive detection [11]. Neural networks provides a method for defining a system too complex to be defined by a simple model, e.g. image recognition and other systems influenced by uncertainty.

A really important parameter concerning the overall capability of the neural network is it’s architecture. The architecture concerns number of layers, number of neurons in each layer, connections between neurons etc. A nematode worm possesses only 302 neurons in total [12]. Still this presumably unintelligent worm is capable of performing complex tasks super computers today have troubles with. This is due to the complexity of the yet unknown inner mechanisms and architecture of a worm’s biological neural network. As stated before, ANNs are inspired from the biological networks in human brains. However, the extremely complex brain is still not fully understood even by scientists who have devoted large part of their professional life investigating the human brain. ANNs architecture are therefore just a mere sketch of the complex biological version. Still, through the 4th industrial revolution we are experiencing today, new methods, algorithms and frameworks emerge rapidly [13].

Machine learning applications have achieved state-of-the-art performances in multiple disciplines using ANN. Google’s *AlphaGo* has beaten the worlds best human Go player, and is arguably the strongest Go player in history [9]. *ImmerEye* by Microsoft uses machine learning to develop image diagnostic tools in order to detect tumors etc. [14]. Machine learning approaches are also believed to have a dramatic impact in the fields of economics [15] in the short future. Thus, it is safe to say that machine learning will, at some extent, impact the majority of the modern generation.

C. Contributions

This paper investigates a method for transfer learning in underwater domains. Existing methods have not to a large extent been tested for use in underwater domains. In the presented work, experiments are conducted for two different datasets obtained in an underwater environment. Large datasets required for machine learning applications can be expensive and difficult to acquire. Applying transfer learning methods for underwater environments can provide an alternative method for cost-effective and simple dataset generation. This paper provides a collective overview of state-of-the-art frameworks targeting transfer learning topics. Moreover, suggests solutions for reduction of the reality gap in the learning process of machines. The main contribution of the work is the application of a transfer learning framework to vision-based underwater operations.

The outline of the paper follows with Sec. II describing investigated methods involving transfer learning. Sec. III describes the experiment setup and datasets as well as conducted simulations, before the results are presented and discussed in Sec. IV. Lastly conclusions and recommendations regarding further work are presented in Sec. V.

II. RELATED WORK

Transfer learning is a substantial problem in machine learning. A robotic arm can be trained to sort red and yellow cubes. However, such training algorithms often run into problems if the color of the cubes change to blue and green. Or, if the shape changes to triangles, or simply the lightning setting changes. Algorithms trained in a simulated environment often experience a problem when they are applied to real world data. This is referred to as the *reality gap*. Different approaches have been developed to reduce this gap between a simulated environment and the real world. A suggested solution is to train on a variation of simulated environment data. [16] developed an object detector that trained using only simulated data. The paper focused on a robotic arm that would grasp desired objects in a cluttered environment. They found it possible to train the detector to 1.5cm accuracy. The simulator they utilized consisted of randomly rendered images with variation in camera position, lighting conditions, object positions and non-realistic-textures. The objective was to perceive the real environment as just another variation. They demonstrated how their object detector could achieve high enough accuracy when tested in real life even though it only had been trained on in a simulated environment.

A breakthrough within the transfer learning topic arguably came in 2014 when Generative Adversarial Networks (GAN) was introduced [7]. The network consist of a combination of two networks, a generator and a discriminator. The generator aims to produce content, while the discriminator determines the level of authenticity of the content. They learn simultaneously and compete against each other, in what can be described as a zero-sum game. The generator produces samples $x = G(z)$, and the discriminator attempts to determine if the samples are produced by the generator or if they come directly from the

training set. The discriminator produces a probability given by $D(x)$, indicating the probability that x is a real sample rather than a fake sample produced by the generator. The end-goal of GAN is that the discriminator will be unable to distinguish the real samples from the fake and produce a constant probability of 0.5. The discriminator will focus on learning to correctly classify samples as real or fake, while the generator will simultaneously try to generate as real looking samples as possible to fool the discriminator. This model can be highly under-constrained, but there exist several published methods and frameworks solving this.

Coupled Generative Adversarial Network (CoGAN) is a framework for learning joint distributions between individual domains [17]. The model aims to obtain a learning based on the joint distributions between domains rather than learning from corresponding images in different domains. This simplifies the requirements of the datasets, because CoGAN doesn't require corresponding images in the different domains. The framework discovers the joint distribution instead. CoGAN has been applied for color and depth images, as well as on face images with different attributes and demonstrated successfully image transformations between domains.

Based on the CoGAN framework, [18] illustrates a method for unsupervised image-to-image translation. The method learns a joint distribution between individual domains, by assuming there exists a shared-latent space. The shared-latent space assumption assumes a pair of corresponding images in different domains can be mapped in the same latent domain. The authors demonstrated image-to-image translation between two domains without any corresponding images in the training datasets. Moreover, a limitation of the presented translation is a unimodal model due to the Gaussian latent space assumption. A unimodal model means there exist only one peak, i.e. one right answer. Another limitation is possible unstable training due to the saddle point searching problem.

pix2pix uses conditional GAN (cGAN) to learn the translation between domains [19]. Since the release of the framework, a large number of different experiments has been conducted by different people. The framework shows promising results. The downside of pix2pix is the need for correlating image pairs in the source and target domain. A modified version of GAN, CycleGAN, is a method to perform image-to-image translation between domains without paired images in each domain [20]. The independence from paired images as well as wide range of domains CycleGAN has been applied to, are the main reasons why CycleGAN is the contemplated framework for this paper.

A. CycleGAN

A thorough description of the CycleGAN framework can be found in [20]. Moreover, an overall description of the framework and how the cycle consistency is implemented in the framework is summarized here. The image-to-image translation is achieved by adding an additional generator and discriminator. The framework attempts to learn the mapping $y = G(x)$ and $x \approx F(G(x))$, where G and F are two different generators. CycleGAN is one of the recent most successful

approaches to the image domain transformation topic. Introducing $x \approx F(G(x))$ provides an additional loss function, the cycle consistency loss, in addition to the adversarial loss. The adversarial loss is defined with

$$L_{GAN}(G, D_Y, X, Y) = \mathbf{E}_{y \sim p_{data}} \log D_Y(y) + \mathbf{E}_{x \sim p_{data}} \log (1 - D_Y(G(x))), \quad (1)$$

where G is the mapping function attempting to generate images $G(x)$ similar to images in domain Y . D_Y attempts to distinguish between the generated images, $G(x)$, and the real images y .

In order to implement a desired cycle consistent mapping, the cycle consistency loss is added, (2). This loss ensures that for each image, x or y , the original image is reconstructed after the image translation cycle, i.e. $X \approx F(G(X))$ and $Y \approx G(F(Y))$, as previously mentioned.

$$L_{cyc}(G, F) = \mathbf{E}_{x \sim p_{data}} \|F(G(x)) - x\|_1 + \mathbf{E}_{y \sim p_{data}} \|G(F(y)) - y\|_1 \quad (2)$$

The objective in CycleGAN will consequently be a sum of the adversarial loss and the cycle consistency loss, represented with the final loss function

$$L_{CycleGAN}(G, F, D_x, D_y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(G, D_X, Y, X) + \lambda L_{cyc}(G, F). \quad (3)$$

λ determines the relative importance of the two objectives. Notice that the final loss function is represented with two functions for adversarial loss. This is to ensure the losses for mapping between both domains are accounted for. Considering the loss function given by (3), the objective of CycleGAN will be to solve

$$G^*, F^* = \arg \min_{G, F} \max_{D_x, D_y} L_{CycleGAN}(G, F, D_x, D_y). \quad (4)$$

As mentioned, CycleGAN offers unpaired image-to-image translation. Regarding datasets, this provide a great advantage, because datasets can be extracted from already existing data in the industry. The framework can also provide the translation with unlabeled dataset, which means time spent on labeling each element in vast amounts of data can then be avoided.

III. EXPERIMENTAL SETUP

In this section the two contemplated datasets will be introduced. Parameters regarding the training and testing of the CycleGAN framework will also be presented.

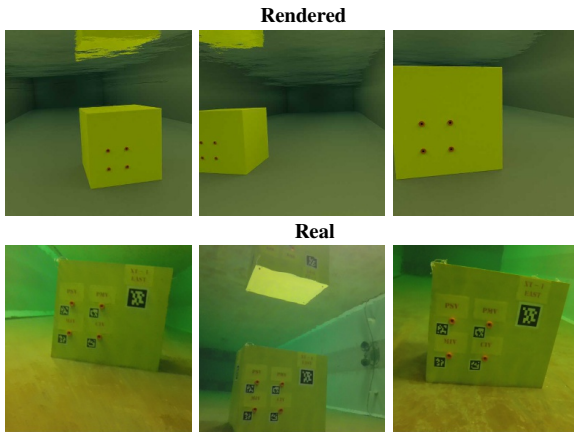


Fig. 1: Dataset 1.

A. Dataset

The datasets that will be used for simulations are two sets containing real and rendered images of a subsea panel. Subsea panels are installed on oil and gas templates on the Norwegian Continental Shelf. The panels are accessed by ROVs for e.g. valve operations and the ROVs operators are totally dependent of good images. In case of autonomous valve operations, automatic systems based on machine learning techniques and the CycleGAN framework is one solution for image characterization. The datasets contains no corresponding images in the training sets, meaning there exist no specific image for one domain corresponding to another image in the other domain. The datasets are also unlabeled. The framework is therefore required to map the features between the domains without being told the correspondence between them. Dataset 1 contains images of a subsea panel placed in the marine cybernetics laboratory (MC-lab) at NTNU [21], as well as rendered images of the same environment. This dataset contains four different directories.

- **trainA**: Containing 4868 rendered .jpg images of the subsea panel at the bottom of the MC-lab.
- **trainB**: Containing 2947 .jpg real images of the subsea panel at the bottom of the MC-lab
- **testA**: Containing 132 rendered .jpg images of the subsea panel at the bottom of the MC-lab
- **testB**: Containing 118 .jpg real images of the subsea panel at the bottom of the MC-lab

The images are taken from a videostream filming the subsea panel at different angles, while the rendered images are rendered using the software blender. Fig. 1 represents image examples taken from the dataset.

Dataset 2 contains real and rendered images of a subsea panel placed in the fjord outside Trondheim. These are images taken at a more realistic setting, which naturally contains more noise than the images from the laboratory. The rendered images are taken from a computer aided design (CAD) model

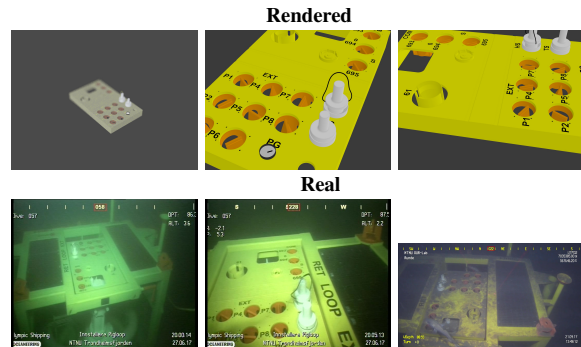


Fig. 2: Dataset 2.

where angles, distance and different noise patterns are altered to ensure the dataset contains variance. The images can be seen in Fig. 2. The dataset is split into 4 directories with

- **trainA**: Containing 1786 rendered .jpg images of the subsea panel.
- **trainB**: Containing 406 .jpg real images of the subsea panel at the bottom the fjord.
- **testA**: Containing 200 rendered .jpg images of the subsea panel.
- **testB**: Containing 46 .jpg real images of the subsea panel at the bottom of the fjord.

Both datasets represents an underwater environment, however at different extent. Dataset 1 is from a laboratory and the images are characterized by clear water and light conditions, not unlike a surface environment. Dataset 2 are more characterized by a typical underwater environment. The environment is dark, reflection from light source occurs and fouling are present at a representative amount of the images. Another reoccurring issue in underwater environments is marine snow which leads to occluded images. Moreover, the images in this dataset are taken from inside a fjord, which results in marine snow being almost non-existent with relative clear images due to calm water.

B. Framework

Dataset 1 discussed above includes images with size 256×256 . When training on this network, both load size and crop size of the framework are set to 256×256 in order to maintain the resolution of the input images. Dataset 2 includes images with different image sizes. The rendered images includes images in sizes 1080×980 and 1080×800 . The real images varies between 1920×1080 and 720×576 . When training on this dataset, the load size is set to 286×286 and crop size 256×256 . The different sizes of input images are therefore coped with by loading all images into the framework with equal size before they are cropped.

The model trains for 200 epochs, which represents going through the entire dataset 200 times. The training is conducted with a constant learning rate of 0.0002 for the first 100 epochs, before decaying towards 0 for the last 100 epochs. For all

simulations, λ from equation 3 is set to $\lambda = 10$. The model is saved every 3000 iterations. In order to keep track of the progress during training, examples of the current state are generated every time the model is saved. The discriminator network architectures are 70×70 PatchGAN networks. The regular GAN discriminator maps from a 256×256 input to a scalar output to determine real or fake. In comparison, the PatchGAN discriminator maps from a 256×256 input to a $N \times N$ network of X outputs, where X_{ij} signifies whether the patch ij in the input image is real or fake. The architecture of the generator networks contains two 2-stride convolutions, nine residual blocks and two fractionally-strided convolutions with stride $1/2$. Further, the framework is trained with a batch size of 3 with batch normalization. The batch normalization ensures that the loss is calculated over the batch and not for each instance. The framework is trained on a *Nvidia GeForce RTX 2080 Ti/PCIe/SSE2* graphics card.

IV. RESULTS AND DISCUSSION

In this section the results from the simulations will be presented. Both datasets have been tested on transfer learning between the two domains and the results vary. The varying results will also be discussed and suggested improvements and solutions will be presented.

A. Results

The results are obtained by testing the framework on the test directories with the trained weights. A part of the obtained results are depicted in Fig. 3. The figures includes six original input images from the rendered domain and the corresponding generated output. For both datasets. Testing has been conducted between both domains, in order to see if the framework has correctly mapped the relevant features in the two domains. However, regarding the task of generating datasets for future machine learning applications, the results presented in Fig. 3 would be the most interesting. Proper generation of images in the real world domain enables generation of vast datasets from rendered images. If large and decent datasets are hard to obtain, this method can provide a more cost-effective alternative.

The results are most satisfying for dataset 2. It can be seen from the figures that for dataset 2, the framework manages to transfer the subsea panel into the other domain in a good manner. The overall structure from the input is kept while the domain changes towards the real domain. Regarding dataset 1 the results are not as satisfying. The output drastically deviates from the input. The relative angle between the camera and structure as well as the spatial features are changed in the output relative to the input. The details on the subsea panel also seems to be randomly placed on the panel. This suggests that the mapping between the domains has been unsuccessful. The domains possess some different features, e.g. QR-codes are neglected in the rendered domain. The framework might encounter issues mapping features that is simply non-existent in one of the two domains.

Fig. 4 also illustrates an insufficient mapping between the domains. For both datasets. This figure depicts the results of applying the domain transfer on the real domain and transferring the input images to the rendered domain. Dataset 1 demonstrates the same issues as for the opposite domain transfer, where angles and spatial features of the input images are changed in the domain transfer. This strengthen the theory that the features of the domains has not been properly mapped due to the low level of details in the rendered images. The domain transfer for dataset 2 seems to encounter much of the same issues. The input images includes parts of the structure that are not included in the rendered domain. The additional structure circumventing the subsea panel as well as the robotic manipulator in the images are unknown to the rendered domain. Consequently, the framework have problems transferring these features into the rendered domain where they are completely absent.

As previously stated, CycleGAN compares the original image to a reconstructed image in order to calculate the cycle consistency loss. This is depicted in Fig. 5. The figure depicts the input image fed to the framework and the output is the corresponding image in the other domain. The reconstructed image is generated by taking the output image as input and then transferred back to the first domain. The reconstructed images represents the input very well. Notice that for the second image line, the four orange dots are almost gone in the reconstructed image. This demonstrates that the framework perceives these features as non-important. The orange dots are the only features on the rendered subsea panel with information about where the QR-codes should be placed. If these features are seen as non-important it could explain why the generated subsea panels from Fig. 3a are often flipped.

B. Discussion

The results are promising and illustrates a decent mapping between the domains, especially for dataset 2. However, less satisfactory results were also obtained, which indicates that the feature mapping may not be as robust as desired. It should be noted that the level of details on the CAD models could be a reason. The CAD model in dataset 1 illustrates a yellow box with four orange dots. The different QR-codes that are present on the real model are neglected in the CAD model. This may confuse the framework when it attempts to map these exact features between the two domains. This might also be a reason why the generated images of the subsea panel often is flipped for this dataset. The largest QR-code are placed in the upper right corner of the real model. See Fig. 1. However, for the constructed images, it seems randomly placed in either of the upper corners. Placing QR-codes on the rendered model could help ensure that the placement is perceived as a more important feature to the framework. This dataset is obtained in the MC-Lab at NTNU, and the images are not characterized by the dark underwater environment. It is therefore believed that the issues of mapping the features between the domains, is due to the lack of details in the CAD model rather than the fact that the domains are underwater. Moreover, we do

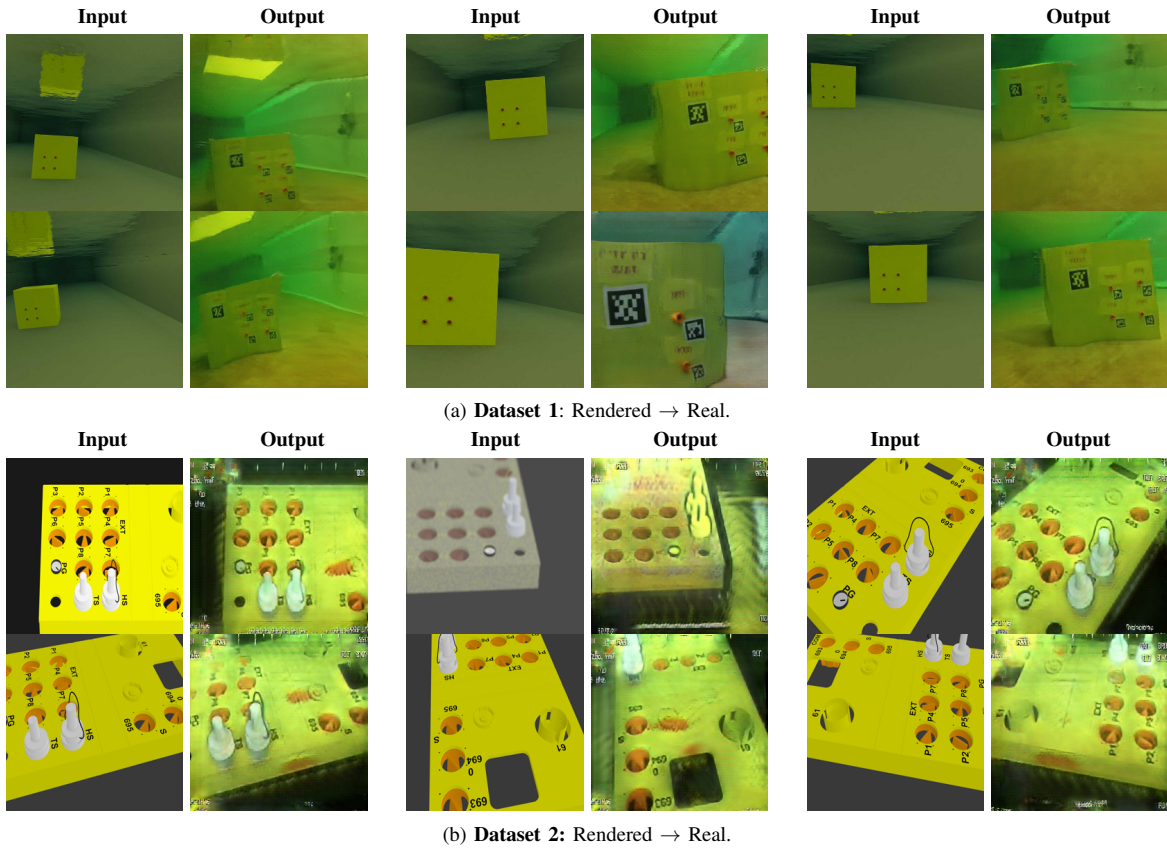


Fig. 3: Results from generating images in the real domain with rendered images as input.

not know exactly how the features are mapped in the neural network, which makes it difficult to determine how much such changes would improve the framework. It is also unknown if they would improve the results at all.

The level of details are increased for dataset 2. However, even though the CAD model is relatively detailed, it only includes the panel itself. The real images shows a panel placed on a larger subsea structure with a manipulator often occurring in the images as well. The additional subsea structure around the panel is non-existent in the CAD-model, which causes some mapping issues. Constructing images of the panel from a distance cause varying results due to this missing structure in the CAD-model. When constructing close up images of the panel on the other hand, the framework performs very well. On the close up images, the level of details are quite similar for the CAD-model and the real images. This provides good circumstances for the framework to map the features between the domains. This dataset is also much more characterized by an underwater environment. When the panel is seen from a distance it is perceived blurry, and the lighting becomes a strong feature. Due to the dark environment, a source of light is necessary in order to light up the subsea panel. Fouling on

the structure, distance of the camera, occluding of camera or light source and other factors provides different reflections on the structure and provides a challenging domain to map. Still, the framework is able to represent this light reflection in a good manner.

Overall, the framework performs well. Limitations that occurs in the domain transfers are believed to arrive from different features not being present in both domains, rather than features characterized by underwater environments. Since the framework is able to comprehend with such circumstantial features, the obtained limitations should be possible to improve similarly to surface limitations.

A limitation with the dataset is the absent of marine snow in the images. The environments on the Norwegian Continental Shelf are deeper, darker and more demanding than the datasets presented in this paper. Another limitation is that the results are hard to evaluate with a metric, due to the dataset being unlabeled. A labeled dataset provides a ground truth, which the results can be compared to. Two popular evaluation metrics for the results presented in this paper are the Average precision (AP), which is often used when measuring accuracy of classifiers [22], and the FCN-score used in [19].

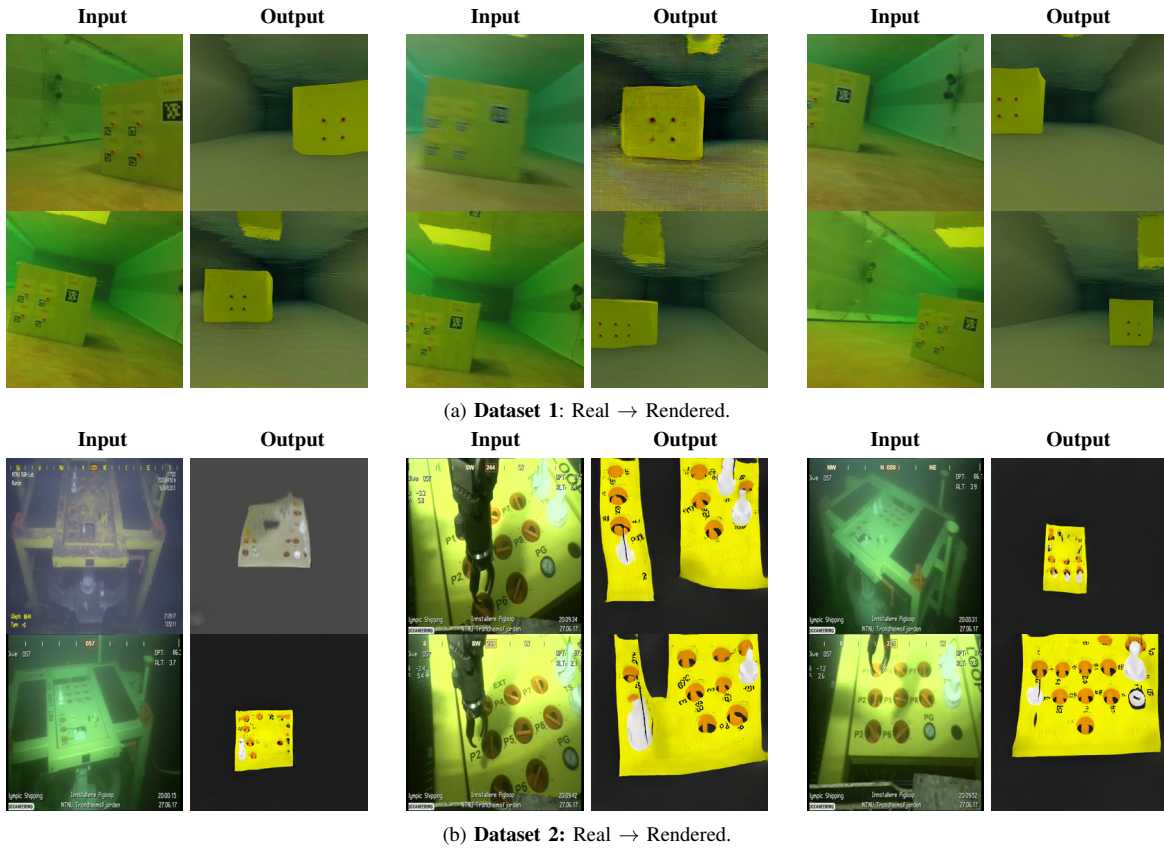


Fig. 4: Results from generating images in the rendered domain with real images as input.

V. CONCLUSIONS AND FUTURE WORK

This paper investigate methods for reducing the reality gap for vision based systems in the underwater segment. Simulations have been conducted on two different underwater datasets in order to apply existing methods at underwater environments. CycleGAN has been used as the contemplated framework. The datasets consist of rendered and real images of two different subsea panels. The framework was trained for 200 epochs on the two different datasets and the results demonstrated a partially successful mapping between the domains. Some results were satisfactory, but less satisfactory results revealed a less robust feature mapping. The framework proved to be able to map features characterized by underwater environments, such as dark images and light reflection. It is therefore believed that increasing the level of details on the CAD models could provide a solution for increasing the robustness of the feature mapping. Moreover, using labeled datasets can also provide possibilities for using evaluation metrics. These issues should be addressed in future work.

ACKNOWLEDGMENTS

Special thanks to MSc. student Daniel Harper for providing the CAD model utilized in dataset 2. Also thanks to MSc. student Fabian Skjølvsvik for editing the images in dataset 2 and setting it up as a functioning dataset.

REFERENCES

- [1] K. Bousmalis and S. Levine, "Closing the simulation-to-reality gap for deep robotic learning," Available at <https://ai.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html>, October 2017.
- [2] ResearchAndMarkets.com, *Underwater Robotics Market Size, Share Trends Analysis Report By Type (ROV, AUV), By Application (Commercial Exploration, Defense Security, Scientific Research), By Region, And Segment Forecasts, 2018 - 2025*. ResearchAndMarkets.com, 2018.
- [3] finbox.io, "Revenue cagr (5y) for apple inc." Available at https://finbox.io/AAPL/explorer/revenue_cagr_5y, November 2018.
- [4] sintef.no, "Ocean space centre får grønt lys fra kvalitetssikrer." Available at <http://www.oceanspacecentre.no/wp-content/uploads/2018/04/Supplerende-analyse-Ocean-Space-Centre-forkortet.pdf>, October 2018.
- [5] I. Schjølvberg, T. B. Gjersvik, A. A. Transeth, and I. B. Utne, "Next generation subsea inspection, maintenance and repair operations," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 434 – 439, 2016, 10th IFAC Conference on Control Applications in Marine SystemsCAMS 2016.

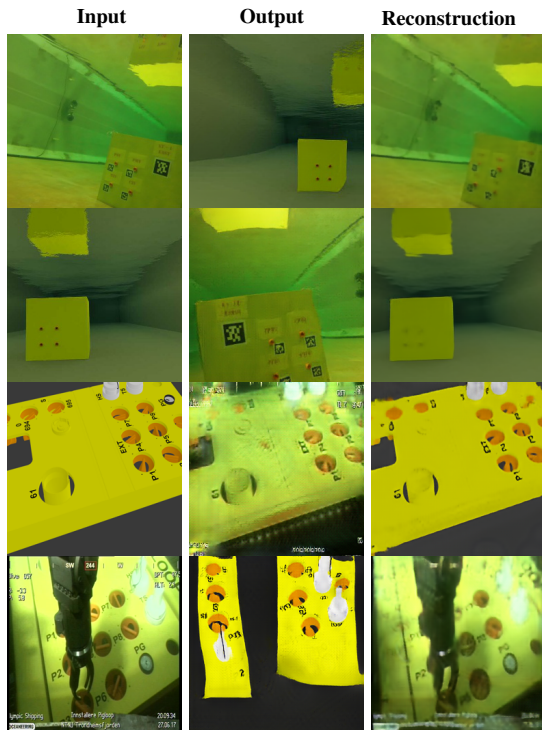


Fig. 5: Input and output with reconstructed image.

- ulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 23–30.
- [17] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 469–477.
- [18] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *CoRR*, vol. abs/1703.00848, 2017.
- [19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.
- [20] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR*, vol. abs/1703.10593, 2017.
- [21] "Marine cybernetics teaching laboratory," Available at <https://www.ntnu.edu/imt/lab/cybernetics>.
- [22] J. Hui, "map (mean average precision) for object detection," Available at https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, March 2018.
- [6] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, April 1980.
- [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [8] V. Zwass, "Neral network," Available at <https://academic.eb.com/levels/collegiate/article/neural-network/126495>, November 2018.
- [9] "Alphago," Available at <https://deepmind.com/research/alphago/>, November 2018.
- [10] R. Almodfer, S. Xiong, M. Mudshb, and P. Duan, "Multi-column deep neural network for offline arabic handwriting recognition," in *Artificial Neural Networks and Machine Learning – ICANN 2017*, A. Lintas, S. Rovetta, P. F. Verschure, and A. E. Villa, Eds. Cham: Springer International Publishing, 2017, pp. 260–267.
- [11] H. Wang, Y. Li, Y. Yang, S. Hu, B. Chen, and W. Gao, "Study of artificial neural network on explosive detection with pftna method," in *IEEE Nuclear Science Symposium Conference Record, 2005*, vol. 1, Oct 2005, pp. 471–473.
- [12] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, "The structure of the nervous system of the nematode caenorhabditis elegans," *Philosophical transactions of the Royal Society of London*, vol. 314, no. 1165, pp. 1–340, 1986.
- [13] B. Marr, "What everyone must know about industry 4.0," Available at <https://www.forbes.com/sites/bernardmarr/2016/06/20/what-everyone-must-know-about-industry-4-0/2941b856795f>, June 2016.
- [14] "Project innereye – medical imaging ai to empower clinicians," Available at <https://www.microsoft.com/en-us/research/project/medical-image-analysis/>, 2018.
- [15] S. Athey, "The impact of machine learning on economics," Available at <https://www.nber.org/chapters/c14009.pdf>, January 2018.
- [16] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from sim-

Article 2

Dynamic Positioning of and Underwater Vehicle using Monocular Vision-Based Object Detection with Machine Learning

Martin Skaldebø, Bent Oddvar Arnesen Haugaløkken, Ingrid Schjølberg
Submitted to the *OCEANS 2019 Seattle Conference & Exposition*. Seattle, USA.

Dynamic Positioning of an Underwater Vehicle using Monocular Vision-Based Object Detection with Machine Learning

Martin Skaldebø
Dept. of Marine Technology
NTNU
Trondheim, Norway
martin.b.skaldebo@ntnu.no

Bent Oddvar Arnesen Haugaløkken
Dept. of Marine Technology
NTNU
Trondheim, Norway
bent.o.arnesen@ntnu.no

Ingrid Schjøberg
Dept. of Marine Technology
NTNU
Trondheim, Norway
ingrid.schjolberg@ntnu.no

Abstract—Sonar and camera are two widely used sensors in the underwater segment. Moreover, optical based systems provide higher spatial and temporal resolution than their acoustic counterpart. In this paper, a dynamic positioning system for a small-class ROV relative to an object detected by a monocular camera will be presented. The object detection will be performed by the state-of-the-art object detector YOLOv3 trained on a dataset representing the relevant known object. In order to perform dynamic positioning based on 2D images a scaling will be used to extract the spatial features from the images. The entire system is able to perform at real-time which is essential for a dynamic positioning procedure.

Index Terms—Underwater robotics, object detection, autonomy, data augmentation, dynamic positioning

I. INTRODUCTION

Subsea inspection, maintenance and repair (IMR) operations are often identified with the offshore oil and gas industry, but lately also highly relevant for deep sea mining and aquaculture. IMR operations are commonly executed by underwater vehicle-manipulator systems (UVMS), which today rely heavily on humans. Increasing the level of autonomy and optimizing the human-robot interaction in these operations can potentially reduce costs and increase safety [1].

In teleoperated systems, the human operator is aided by visual and sensory feedback in order to assess the situation, make decisions and remotely execute tasks. The same conditions apply for autonomous systems, only then the system itself needs to conduct the operations without a human in the loop, increasing the demands regarding the sensory systems and implemented software. In the object detection aspect of autonomous operations, sonars and cameras are two widely used sensors [2]–[4]. Acoustic sonars have for a long time been a preferable sensor in underwater systems. However, recent technological advance within camera systems and the use of visual aid proves that camera systems have potential for short range navigation. Moreover, visual aided systems may provide systems with higher spatial and temporal resolutions than the acoustic counterpart [5].

It is not straightforward to use camera systems in underwater environments, especially when paired with robotic systems

during semi- or fully autonomous operations. The underwater scene is considered one of the most difficult to perform optical detection and recognition of objects and patterns. This is because underwater image quality heavily depends on absorption and scattering of light [6] [7]. With regards to the operation at hand, underwater object detection is typically performed with a specific object in mind, which might be fully visible, or either partially or fully obscured by other objects. With recent technological advances and lowered costs of graphical processing units and cameras, object detection can be performed both quickly and reliably, hence making the method suitable in conjunction with autonomous control application. Simultaneously, the same type of development has been seen in commercial underwater vehicle products, such as the BlueROV2, which allows customization and testing of new hardware with user-made software. Incorporating object detection methodologies for underwater vehicles enables more autonomous functionality in underwater robotics, such as tracking of objects during IMR or visually aided manipulation operations, whether it is used in exploration operations, within the marine oil and gas or the aquaculture industry.

This paper will investigate methods for efficiently labelling of large datasets and training generated dataset based on the state-of-the-art object detection framework YOLOv3 [8]. The object detection will be used to locate a known object with a monocular camera mounted in a BlueROV2 underwater vehicle. Moreover, spatial features will be extracted from the object detection in order to estimate the relative distance between object and ROV. A dynamic positioning (DP) system is designed to keep the vehicle at a desired relative position and orientation to the object. The main contributions of this paper are as follows:

- Collect a large dataset of the object and provide underwater object detection using machine learning
- A method for efficiently labeling of large datasets
- Extraction of spatial information from monocular camera underwater images
- A DP-system where the vehicle has a desired position

relative to a known object, where position of the object is extracted from monocular camera

- Demonstration of methods and concept in a laboratory pool

The paper is structured as follows. Section II provides some background information on visually aided control using underwater vehicles and presents related work on object detection topics. Section III presents the dataset and how the labeling and scaling procedures are conducted as well as how the detector works. Furthermore Section IV describes the motion control system, while the experimental setup is explained along with results in Section V. Lastly, the results are discussed in Section VI and conclusions and suggestions for further work are provided in Section VII.

II. RELATED WORK

Optic-based underwater object detection is a hot topic today, especially within the research community and various subsea industries, but also for hobbyists, where a consistent series of new contributions have surfaced in the last few years. The most common vision-based techniques utilize either monocular (2D) or stereo (3D) vision, while 2.5D methods also have been proposed, mainly consisting of projection algorithms from the 2D image plane to reconstruct 3D environment features [6]. Detection methods range from on edge [9] [10] or color [11] detection, optical flow [12], and techniques relying more on machine learning approaches such as classification [13], salient feature detection [14] or object detection [15] [16]. Color detection simply finds and draws contours around neighboring colored pixels in an image, while edge detection denotes the boundaries of objects. Optical flow is used to track individual pixels or pixel areas, and can be used in combination with the aforementioned methods.

Within machine learning, there is a wide array of various detection methodologies, mainly identified as two different types. These are the region proposals based detectors and the regression and classification based detectors. They both originate from generic object detection as illustrated in Fig. 1. The first follows the traditional object detection procedure, identifying region proposals and classifying the proposals into object categories, also known as Region Proposals Networks (RPN). Such methods consider, at some extent, the same methodologies as the human brain. This method is based on an initial scan of the entire scenario before it is separated into regions of specific interest. The second type follows a classifier-based approach or a regression problem. These methodologies both arrive from the supervised learning branch of machine learning. The branches of machine learning are shown in Fig. 2. Thus, they share the overall objective of supervised learning, that is; learn the mapping from input x to output y , i.e. learn the mapping function $f(\cdot)$ in $y = f(x)$. The main difference is that while classification approaches aim to learn the mapping to a discrete or categorical output, regression approaches aim to learn the mapping to a continuous or numerical output.

A collected review of the most essential methods of both RPN and regression/classification based approaches can be

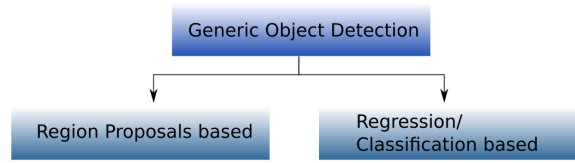


Fig. 1: Object Detection Methodologies

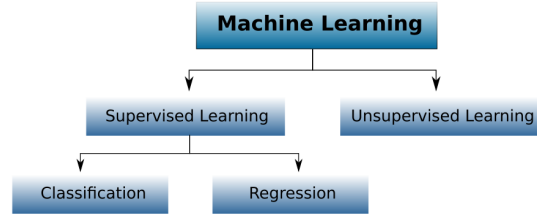


Fig. 2: Machine Learning methodologies

seen in [17]. Some methods worth mentioning are the RPNs R-CNN, Fast R-CNN and Faster R-CNN and the regression/classification based methods Single Shot Detector (SSD) and You Only Look Once (YOLO) versions 1, 2 and 3. R-CNN is region proposals combined with convolutional neural networks, hence the name.

The R-CNN algorithm proposes 2000 regions per image. The algorithm then works with these regions and attempts to classify them in order to locate the object. 2000 regions per image is a lot to process and is computationally expensive. The algorithm uses about 47 seconds per image, which makes it irrelevant regarding real-time detection. A modified version, Fast R-CNN, uses a slightly different approach. Instead of proposing 2000 regions per image, the modified version feeds the input image to a CNN and outputs a convolutional feature map. The proposed regions from the feature map is fed through a regions of interest (RoI) pooling layer and a fully connected layer. Then a softmax layer is used in order to predict the class of the object as well as the regions for the bounding boxes. As for R-CNN, Fast R-CNN uses selective search, which is a slow and time-consuming process. Faster R-CNN is a further modification of the algorithm to circumvent the selective search algorithm. Faster R-CNN uses a separate network instead of selective search in order to predict the region proposals. This results in a much faster network. Faster R-CNN is the only network out of these three that is fast enough to be applicable for real-time object detection [18]–[20].

SSD uses a classification/regression based approach and, hence, does not require object proposals, and encapsulates all computation in a single network [21]. Consequently the algorithm is fast and suitable as a real-time object detector. SSD also proved to be more accurate than the YOLO versions available at the time. However, new versions of YOLO has emerged since then. The version YOLOv3 has proven to be

just as accurate as SSD, however three times faster [8]. Still YOLOv3 is not as fast as its predecessor, where YOLOv2 could run on a Titan X at 45 FPS while YOLOv3 is limited to about 30 FPS. This is due to the increase complexity of the underlying network called Darknet. YOLOv2 used Darknet-19 consisting of a total of 30 layers, contained originally a 19-layer network with additional 11 layers for object detection. YOLOv3’s network Darknet-53 contains 53 layers trained on ImageNet with 53 more layers for object detection. This gives a network with 106 layers. This increased complexity is the cause of decreased speed as well as increased accuracy. Darknet-53 incorporated some new important elements which Darknet-19 did not contain, such as residual blocks, skip connections and upsampling. See Table I for the Darknet-53 layers. Another issue YOLOv3 has addressed is the fact that previous versions have struggled with detecting small objects. In YOLOv3, predictions are made at three different scales, and after each detection, layers are upsampled. The upsampling helps the network learn fine-grained features, which are advantageous for detecting small objects.

As mentioned previously, pairing visual tracking methodologies of objects with underwater vehicles has a tremendous potential in autonomous underwater operations. One of the most famous and successful combination of this sort was demonstrated by the underwater vehicle named SAUVIM [22]. The object detection phase, which they considered to be the most difficult part of the project, was threefold. Image sonar and DIDSON sonar were used at long- and mid range detection, while for the actual manipulation tasks, video cameras were used in collaboration with ultrasonic motion trackers. Further development of the SAUVIM project was later presented by the TRIDENT and MARIS projects [23], [24]. The TRIDENT project demonstrated the first multi-purpose object search and recovery strategy. Similar to the

TRIDENT project, the MARIS project stands out as one of the recent most promising projects regarding autonomous underwater manipulation. Compared to the TRIDENT project, the MARIS project improved the vision system. Both projects employed stereo cameras where the MARIS project improved the detection algorithm to cope with partial occlusions of the object.

Some of the other work related to vision-aided robotic control applications can be found in [25]–[29]. The focus reported by these articles are either solely related to underwater object detection, positioning using vehicles and manipulation of detected objects using manipulator arms, or a combination of these. A more in-depth review article on this topic is presented in [6], where recent developments in machine learning methodologies can be seen in [17].

III. DATA AUGMENTATION

In the presented work, the dataset applied represents a known object in an underwater environment in the Marine Cybernetics Laboratory (MC-Lab) at the Department of Marine Technology at the Norwegian University of Science and Technology (NTNU). The dataset includes images retrieved in two stages. The first stage was to retrieve by recording the objects at close range with a monocular camera attached to a robotic manipulator. The second stage was to retrieve by manually controlling the ROV while recording the environment with the object. For both cases a Raspberry Pi Camera V2.1 was used. Specifications regarding pixel size and field of view of the camera are given in Table II. The two recordings were split into image sets to generate the dataset.

A. Labeling

The collected dataset contains 7071 images. Such a vast dataset provides comprehensive work regarding labeling the images. Therefore, methods for efficiently labeling the images were investigated. A popular method for managing large datasets is crowdsourcing, where a task is distributed to numerous participants for analyses. A reoccurring challenge is the unknown reliability of the participants [30] [31]. Our labeling scheme enables safe labeling of a large dataset with over 7000 images in mere hours, where every label is verified by the user in order to ensure the liability of the labels. The labeling process is split in two steps.

- Initial labeling of images using color detection.
- Correcting of wrong/bad labels.

The considered object is characterized by a clear orange color. This made it possible to separate the relevant colors of the object in order to detect the object. The contemplated images were transformed from the RGB space to the hue-saturation-brightness value (HSV) in order generate a more straightforward color map. Such a transformation can be seen in Fig. 3. After the initial label generation using the color scheme, the labels are regulated by the user operating an interactive interface. Utilizing HSV images ensures that the object becomes apparent in the image and simplifies the color separation. The values characterizing HSV images also

TABLE I: Darknet-53

| | Type | Filters | Size | Output |
|----|---------------|----------|-----------|-----------|
| 1x | Convolutional | 32 | 3 x 3 | 256 x 256 |
| | Convolutional | 64 | 3 x 3 / 2 | 128 x 128 |
| | Convolutional | 32 | 1 x 1 | 128 x 128 |
| | Convolutional | 64 | 1 x 1 | |
| | Residual | | | |
| 2x | Convolutional | 128 | 3 x 3 / 2 | 64 x 64 |
| | Convolutional | 64 | 1 x 1 | 64 x 64 |
| | Convolutional | 128 | 3 x 3 | |
| | | Residual | | |
| 8x | Convolutional | 256 | 3 x 3 / 2 | 32 x 32 |
| | Convolutional | 128 | 1 x 1 | 32 x 32 |
| | Convolutional | 256 | 3 x 3 | |
| | | Residual | | |
| 8x | Convolutional | 512 | 3 x 3 / 2 | 16 x 16 |
| | Convolutional | 256 | 1 x 1 | 16 x 16 |
| | Convolutional | 512 | 3 x 3 | |
| | | Residual | | |
| 4x | Convolutional | 1024 | 3 x 3 / 2 | 8 x 8 |
| | Convolutional | 512 | 1 x 1 | 8 x 8 |
| | Convolutional | 1024 | 3 x 3 | |
| | | Residual | | |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

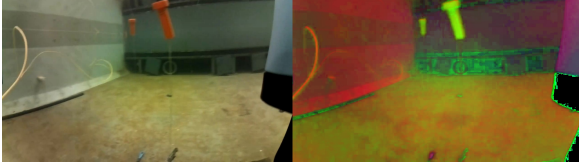


Fig. 3: RGB to HSV transformation of the underwater image

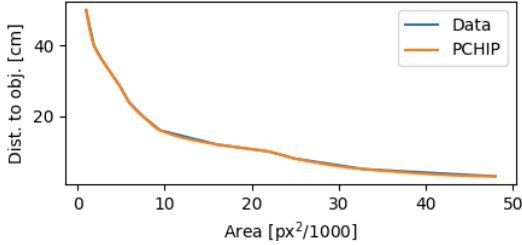


Fig. 4: Visual representation of the scaling function. Area in $\text{px}^2/1000$ is plotted against the corresponding distance to the object in cm. The original measured data is visualized with a blue line and the final scaling using PCHIP is visualized in orange.

provide a simpler spectrum to analyze compared to RGB values.

B. Spatial scaling

In object detection schemes, real-time pose of the object is generally of interest, thus stereo cameras are often used [23] [24] [32]. Stereo camera systems employ two or more lenses in order to simulate humans binocular vision, thus capturing spatial features in images to get a 3D representation. Moreover, in machine learning applications where labeled data is needed, 3D images could be complicated to process and labeling of such images are difficult. Processing standard monocular camera images and labeling ordinary RGB 2D images is much less complicated. However, this requires alternative methods for extracting spatial features.

A scaling function is designed in consideration of the spatial features in the system. The spatial features are important in order to estimate the position of the object relative to the ROV. The scaling function involves scaling the area of the bounding boxes to the corresponding distance to the object. The scaling is performed by manually by measuring different distances to the object and registering the area of the detection bounding box. The final scaling function is generated by using piecewise cubic hermite interpolation polynomials (PCHIP) at the registered values. A visual representation of the scaling is depicted in Fig. 4

Values for (y,z) position of the object are also extracted as pixel values from the image frame. Furthermore the values are transformed into distances by (6) and (7), where the angles ψ and θ are retrieved from (3) and (4), and x is the distance to

TABLE II: Specifications of the Raspberry Pi Camera V2.1

| Parameter | Definition | Value |
|-------------|---|--------------|
| FOV_w | Field of view in horizontal plane | 62.6 [deg] |
| FOV_h | Field of view in vertical plane | 48.8 [deg] |
| C_w | Total pixel width of camera frame | 640 [px] |
| C_h | Total pixel height of camera frame | 480 [px] |
| $P_{obj,w}$ | Object pixel position in width direction | 0 - 640 [px] |
| $P_{obj,h}$ | Object pixel position in height direction | 0 - 480 [px] |

the object retrieved from the scaling function depicted in Fig. 4. The remaining variables in the equations are explained in Table II.

C. Object Detector

The object detector is based on the YOLOv3 algorithm discussed in Section II. The algorithm trains for 5000 iterations with a batch size of 64 and subdivision set to 16. An entire batch is considered between every update of the weights in the neural network. After training on the contemplated dataset of the object in the MC-lab, the object detector performs well and achieves an average precision (AP) of 97.9%, calculated with a threshold of 50%. The AP value is incredibly high and proves that the detection of the object is successful. However, the high AP has to be considered with some constraints. The algorithm was validated on a subset of the entire dataset, meaning the validation set embodies almost identical features as the training set. Consequently, a very high AP can potentially point to a overfitted model just as well as a good one.

The final updated weights from the training procedure are further used in the detection procedure. An algorithm is designed with an image frame as input and coordinates and size of the detected object as output. An outline of the detection algorithm is depicted in Fig 5, where the numeric values of (y,z) represents the center position, where $(0,0)$ is the top left corner. Numeric values for (w,h) represents width and height of the bounding box. All values are given in pixel value. Total pixel dimensions of the camera frame can be found in Table II. Six parameters are defined in the table representing horizontal and vertical field of view of the camera, total pixel height and width of the camera images and the allowed pixel position of the object within the camera image. The algorithm assumes that there exists maximum one object per frame. However, the detector itself can detect several objects per frame, meaning that it is easy to modify the algorithm if tracking multiple objects is desired in the future. The image frames are sent from the camera feed and transformed into OpenCV image objects before they are sent as input to the detection algorithm. In the experiments the system is run on a HP Laptop with Intel Core i7-7700HQ, 16 GB RAM and an NVIDIA GeForce GTX 1060 (6 GB GDDR5 dedicated) graphics card. With this setup, the detection can perform at approximately 30-40 fps for 1080x720 resolution video, which provides real-time compatibility.

IV. MOTION CONTROL SYSTEM

This section describes the motion control system (MCS), i.e. the navigation, guidance and control system, that has been

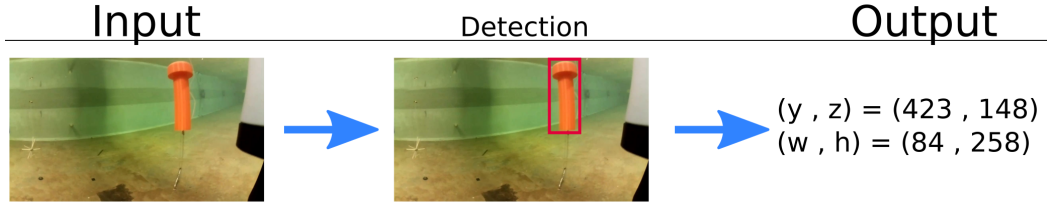


Fig. 5: Outline of the detection algorithm

developed for the unmanned underwater vehicle (UUV) for use in experimental testing. The objective is to keep a fixed position and heading angle of an underwater vehicle through dynamic positioning (DP) relative to an object. It is important to note that the MCS only performs during successful detection of the object, and that the vehicle should use a lower level MCS when the object cannot be detected. However, the navigation, guidance and control system provided below provides low position errors even without a lower level MCS, and is highly advantageous in scenarios where such a system is either non-existent or unreliable. A simple Kalman filter is used for estimating the vehicle's velocity and a sliding mode controller based on the velocity is responsible for controlling the vehicle. Automatic pitch and roll proportional-integral-derivative controllers stabilize the vehicle in roll and pitch. Consequently, roll and pitch motions are handled by a lower level inertial navigation system (INS), and are omitted in the MCS presented here. All forces and moments are handled by a thrust allocation system in order for the thrusters to produce the correct amount of torque.

A. Navigation System

The navigation system only considers the kinematic model of the underwater vehicle relative to the detected object. The kinematic model represents the vehicle's states, which are defined by its pose $\eta = [p^T \theta^T]^T$ and velocity $\nu = [v^T \omega^T]^T$. Vehicle position and Euler angle orientation are described by the vectors $p = [x, y, z]^T$ and $\theta = [\psi]^T$, expressed in the object frame. Linear and angular velocity are defined as $v = [u, v, w]^T$ and $\omega = [r]^T$, expressed in the vehicle's body-frame. Due to the nature of monocular cameras, the position and velocity of the vehicle are valid only when the object is detected, and are defined as

$$\eta = [x \ y \ z \ \psi]^T + w_s^T \quad (1)$$

$$\nu = [u \ v \ w \ r]^T, \quad (2)$$

where the term w_s represents Gaussian distributed white noise. In order to find the values for η , we first calculate

$$\psi = \frac{FOV_w}{C_w/2} \cdot P_{obj,w} - FOV_w \quad (3)$$

$$\theta = \frac{FOV_h}{C_h/2} \cdot P_{obj,h} - FOV_h, \quad (4)$$

where ψ is the heading angle and θ is the roll angle relative to the object, and where the rest of the parameters have been defined in Table II. Next, the vehicle's distance to the object x is retrieved from the scaling function estimated through the PCHIP function $S(A)$, depicted in Fig. 4. The y - and z -distance relative to the object are then found by exploiting the geometrical relations through x and (3)-(4). The position estimation procedure can then be written as follows

$$x = S(A) \quad (5)$$

$$y = x \sin(\psi) \quad (6)$$

$$z = x \sin(\theta). \quad (7)$$

As previously mentioned, a Kalman filter is implemented to estimate the vehicle's velocity ν .

B. Guidance System

The guidance system generates appropriate references for the control system, and is here based on reference velocities. First, the desired position must be defined, and the velocity reference can be derived from this by a simple technique, similar to [33]. The desired position η_d is written as

$$\eta_d = [x_d \ y_d \ z_d \ \psi_d]^T, \quad (8)$$

where x_d, y_d, z_d, ψ_d correspond to the desired distance to the object in x -, y - and z -direction and heading. The reference velocities ν_r is used as a feedback to increase the convergence to the desired position and heading angle of the vehicle, and are calculated as

$$\nu_r = -\gamma(\eta_d - \eta), \quad (9)$$

where γ in (9) is the task reference gain matrix.

C. Control System

The control system utilizes a sliding mode controller (SMC) in order to make the states of the vehicle converge to the desired values by controlling them to a sliding manifold with global exponential stability properties. Similar to [33], the manifold is chosen as

$$s = (\nu_r - \nu) + \Lambda \int_0^t (\nu_r - \nu) d\tau, \quad (10)$$

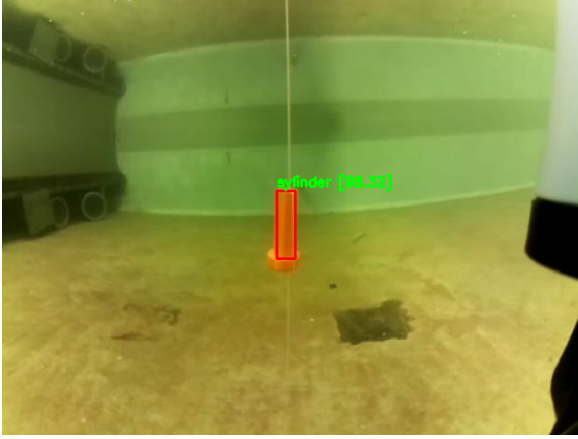


Fig. 6: An image from the camera of the vehicle, showing a successfully detected cylinder by the object detection algorithms, represented by a bounding box (red) and a confidence of 98.32% (green).

where Λ is the integral gain matrix, and where s is globally exponentially stable if $\Lambda > 0$. Finally, the control law [34] [33] is then given as

$$\tau = K_D s + \hat{g}(\Theta) + K_S \text{sat}(s, \epsilon). \quad (11)$$

In (11), $K_D > 0$ and $K_S > 0$ are gain matrices. By assuming that the vehicle is neutrally buoyant and that velocities will be small, restoring forces and moments represented by $\hat{g}(\Theta)$ can be omitted. Furthermore, the function $\text{sat}(s, \epsilon)$ refers to a saturation function of s with lower and upper bound of $\pm\epsilon$, and replaces the signum function to avoid chattering [33] [35].

V. EXPERIMENTAL TESTING

In the experimental testing, a BlueROV2 has been used, which is a small-sized ROV. The vehicle is neutrally buoyant, has six degree of freedom and runs the robotic operating system (ROS) framework for message communication. It is equipped with a monocular camera in front of the vehicle, where the direction of which it points coincides with the heading angle of the vehicle. A small information scheme representing some of the main features of the vehicle can be seen in Table III, and the vehicle itself is depicted in Fig. 7. Experiments have been conducted in the Marine Cybernetics Laboratory (MC-lab) at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The water tank in the MC-lab where the experiments have been conducted is depicted in Fig. 8.

The experiment shows that the vehicle performs DP relative to the object, and is controlled by a sliding mode controller (SMC) based on velocity estimates by a Kalman filter (KF). The KF incorporates the estimated position data of the object through a trained model of the object, as described in Section III. A successful detection during the dynamic positioning can



Fig. 7: The BlueROV2 underwater vehicle [36]



Fig. 8: Water tank in the Marine Cybernetics Laboratory. Dimensions: L x B x D = 40m x 6.45m x 1.5m.

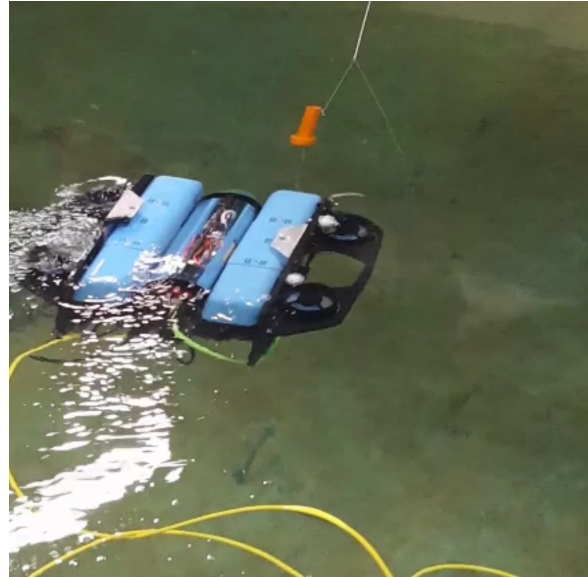


Fig. 9: The Bluerov2 performing dynamic positioning during an experiment in the pool in the Marine Cybernetics Laboratory.

TABLE III: Bluerov2 specifications

| Parameter | Value |
|------------------|--------------------------------|
| L x H x W | 457 [mm] x 254 [mm] x 575 [mm] |
| Weight in air | 11.5 [kg] |
| Weight submerged | 0 [kg] |
| Thrusters | T-200 |
| Battery | 14.8 [V], 10 [Ah] |
| Depth rating | 100 [m] |
| Camera | Raspberry Pi Camera V2.1 |
| Onboard Computer | Raspberry Pi 3B and Navio2 |

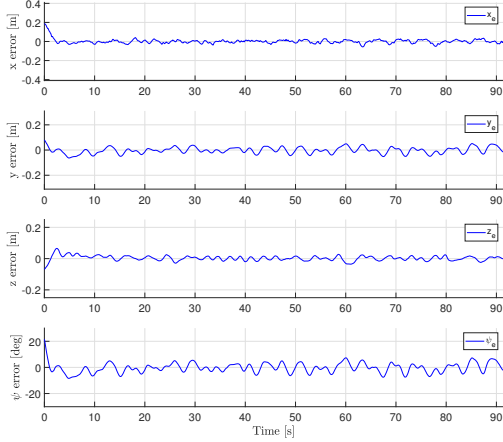


Fig. 10: Vehicle error position (blue) in x-, y-, z-direction [m] and heading angle ψ [deg] from top to bottom, respectively.

be seen in Fig. 6, and the Bluerov2 and object during the experiment is depicted in Fig. 9. This experiment was run for approximately 90 seconds, and the results from this test can be seen in figures 10 and 11. Position and heading errors can be seen in Fig. 10, while velocity and angular velocity errors are represented in Fig. 11.

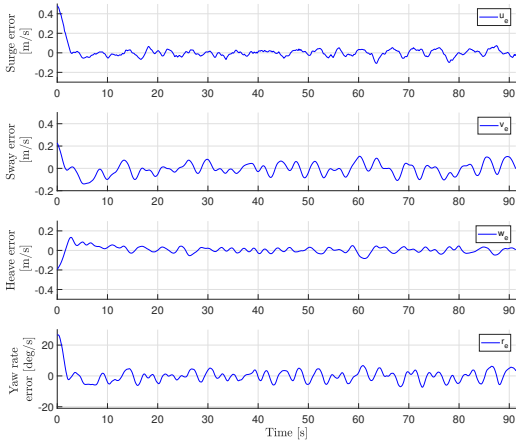


Fig. 11: Vehicle velocity error in x-, y-, z-direction [m/s] and heading angle ψ [deg/s] from top to bottom, respectively.

TABLE IV: Root Mean Square Error for position and velocity of the vehicle relative to the object

| RMSE for position | Value | RMSE for velocity | Value |
|-----------------------------------|-----------|-------------------|-------------|
| RMSE _x | 0.024 [m] | RMSE _u | 0.060 [m/s] |
| RMSE _y | 0.025 [m] | RMSE _v | 0.056 [m/s] |
| RMSE _z | 0.016 [m] | RMSE _w | 0.037 [m/s] |
| RMSE _{ψ} | 3.7 [deg] | RMSE _r | 4.2 [deg/s] |

TABLE V: Gains and parameters used for reference velocity generation and tuning the sliding mode controller.

| Parameter | Value |
|-----------|---|
| γ | [2.5 2.5 2.5 0.3] |
| Λ | [0.02 0.02 0.12 0.2] |
| K_D | [$7.2 \cdot 10^{-4}$ $7.2 \cdot 10^{-7}$ $4.5 \cdot 10^{-3}$ $3.6 \cdot 10^{-6}$] |
| K_S | [$3 \cdot 10^{-3}$ $3 \cdot 10^{-6}$ $1.9 \cdot 10^{-2}$ $1.5 \cdot 10^{-5}$] |

The resulting plots in Fig. 11 show that the velocity is tracked with small errors, which results in fast convergence time and maintaining a small error in all lateral directions and in the heading angle, as can be seen in Fig. 10. Root mean square error (RMSE) is presented in Table IV, denoted by a subscript for the respective state. This table shows an RMSE of around 2.5 [cm] in x- and y-direction, 1.6 [cm] in z-direction and 3.7 [deg] for the heading angle.

The gains of the guidance and control system have been tuned to give a more aggressive steering in x-, y- and z-direction through γ in (9). The integral effect in (10) is slightly larger in z-direction and for the heading ψ compared to the x- and y-direction, in order to compensate for drag forces from the tether. Increased integral effect could have been applied for control in x- and y-direction as well, but the values used here were found to be suitable for following the desired reference velocities. The gain matrices K_D and K_S were chosen to be quite low. Furthermore, the unit of the reference velocity is given in [cm/s], which might be the reason for the seemingly small gain parameters in (9). The gains are presented in Table V.

VI. DISCUSSION

As previously described the object detector was verified on a subset of the entire dataset. The associated value of AP, 97.9% should therefore be considered with constraints. Several experiments should be conducted where new datasets can be retrieved for verification of the model. Optimally, several datasets should be retrieved over several experiments to the final training dataset as well, in order to ensure robustness of the model.

If retrieving more data proves difficult in the future, it is also possible to improve the verification with the currently obtained dataset. The dataset can be divided chronologically into groups of 4. The dataset can then first be trained on the parts 1, 2 and 3 and verified on part 4. Then a new training process should be conducted on parts 1, 2 and 4 and verification on part 3. Repeat the process until the model has been verified on all parts and calculate the average AP of all processes. This will give a better representation of the AP of the model. If the dataset is large, it can also be divided into several groups than

4. However increasing the number of groups will increase the time consumption of the whole process.

In the experimental testing, the MCS for the object tracking mission starts automatically after the object is first detected. The experimental results presented in Figures 10-11 and Table IV show both good convergence rate and capability in maintaining the desired states, with errors below 2.5 [cm] in all directions and a heading angle error below 4 [deg]. There is some oscillatory behavior

The vehicle experiences some difficulties maintaining a certain position relative to the object, which can be seen from the oscillating behavior in Fig. XYZ. This is best explained by the combination of a delay that arises between image acquisition and when the image is ready for processing, and the time it takes for the vehicle's thrusters to propel the vehicle in the desired direction. Less oscillations were achieved by making the controller slower and to slightly decrease the effect of the integral effect. There is also a slight mismatch between the camera angle and the thrust allocation system, as the camera's angle is tilted approximately 5 to 10 degrees in the direction of a positive heading angle for the vehicle, as can be seen from Fig. 6. This may have contributed some to the error in the heading angle, in addition to the x- and y-direction of the vehicle. Another factor that may have contributed to the errors and the somewhat oscillating behavior in Figures 10-11 can be explained by the refraction of light caused by the dome in front of the camera in Fig. 7, as seen in Fig. 6. Yet, the detection algorithm had no trouble detecting the image, and the ROV nicely converged towards and maintained the desired states, and the effect was found to have negligible impact on the performance of the system. Future work may take the refraction into account by calibrating the camera's intrinsic parameters.

Experiments were also conducted for a moving object, although not documented. The experiments demonstrated that the vehicle managed to maintain the desired relative position and followed the objects movement. The experiments were conducted in a laboratory pool with still water and thus not tested for conditions with constant or alternating currents. However, it is believed that the presented MCS will be able to cope with currents, given proper tuning of gains, particularly the integral terms. Future work may involve experiments for such conditions.

VII. CONCLUSIONS AND FUTURE WORK

This paper presents a dynamic positioning procedure relative to an object of interest using a small-class fully actuated underwater vehicle. The object is detected based on a trained model of a large image dataset that contains the object of interest in an underwater environment, using a monocular camera. Furthermore, the paper presents a powerful labeling procedure of the object within the dataset, and a model trained on these images. Experimental testing results prove the effectiveness of the proposed methods, where a small underwater vehicle performs DP relative to the object with small errors. Further work involves adapting the proposed methods to an

underwater vehicle-manipulator system for simultaneous DP on the vehicle and gripping with the manipulator arm.

REFERENCES

- [1] I. Schjølberg, T. B. Gjersvik, A. A. Transeth, and I. B. Utne, "Next generation subsea inspection, maintenance and repair operations," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 434 – 439, 2016, 10th IFAC Conference on Control Applications in Marine SystemsCAMS 2016.
- [2] Z. Chen, Z. Zhang, F. Dai, Y. Bu, and H. Wang, "Monocular vision-based underwater object detection," in *Sensors*, 2017.
- [3] Z. Chen, Z. Zhang, Y. Bu, F. Dai, T. Fan, and H. Wang, "Underwater object segmentation based on optical features," *Sensors (Basel, Switzerland)*, vol. 18, 01 2018.
- [4] H. Cho, J. Gu, H. Joe, A. Asada, and S.-C. Yu, "Acoustic beam profile-based rapid underwater object detection for an imaging sonar," *Journal of Marine Science and Technology*, vol. 20, no. 1, pp. 180–197, Mar 2015.
- [5] F. Bonin-Font, G. Oliver, S. Wirth, M. Massot, P. L. Negre, and J.-P. Beltran, "Visual sensing for autonomous underwater exploration and intervention tasks," *Ocean Engineering*, vol. 93, pp. 25 – 44, 2015.
- [6] Q. Xi, T. Rauschenbach, and L. Daoliang, "Review of underwater machine vision technology and its applications," *Marine Technology Society Journal*, vol. 51, no. 1, pp. 75–97, 2017. [Online]. Available: <https://www.ingentaconnect.com/content/mts/mts/2017/000000051/00000001/art00009https://doi.org/10.4031/MTSJ.51.1.8>
- [7] Z. Chen, H. Gao, Z. Zhang, H. Zhou, X. Wang, and Y. Tian, "Underwater salient object detection by combining 2d and 3d visual features," *Neurocomputing*, 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0925231219304230>
- [8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.
- [9] Y. He, B. Zheng, Y. Ding, and H. Yang, "Underwater image edge detection based on k-means algorithm," in *2014 Oceans - St. John's*, Conference Proceedings, pp. 1–4.
- [10] M. Narimani, S. Nazem, and M. Loeuipour, "Robotics vision-based system for an underwater pipeline and cable tracker," in *OCEANS 2009-EUROPE*, Conference Proceedings, pp. 1–6.
- [11] Z. Chen, Z. Zhang, F. Dai, Y. Bu, and H. Wang, "Monocular vision-based underwater object detection," *Sensors (Basel, Switzerland)*, vol. 17, no. 8, p. 1784, 2017. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pubmed/28771194https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5580077/>
- [12] H. Madjidi and S. Negahdaripour, "On robustness and localization accuracy of optical flow computation for underwater color images," *Computer Vision and Image Understanding*, vol. 104, no. 1, pp. 61–76, 2006. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S107731420600083X>
- [13] Madjidi, Hossein and Negahdaripour, Shahriar, "On robustness and localization accuracy of optical flow computation for underwater color images," *Computer Vision and Image Understanding*, vol. 104, no. 1, pp. 61–76, 2006. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S107731420600083X>
- [14] Z. Chen, H. Gao, Z. Zhang, H. Zhou, X. Wang, and Y. Tian, "Underwater salient object detection by combining 2d and 3d visual features," *Neurocomputing*, 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0925231219304230>
- [15] H. Qin, X. Li, Y. Zhixiong, and M. Shang, "When underwater imagery analysis meets deep learning: A solution at the age of big visual data," in *OCEANS 2015 - MTS/IEEE Washington*, Conference Proceedings, pp. 1–5.
- [16] M. Moniruzzaman, S. M. S. Islam, M. Bennamoun, and P. Lavery, "Deep learning on underwater marine object detection: A survey," in *Advanced Concepts for Intelligent Vision Systems*, J. Blanc-Talon, R. Penne, W. Philips, D. Popescu, and P. Scheunders, Eds. Springer International Publishing, Conference Proceedings, pp. 150–160.
- [17] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2019.
- [18] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.
- [19] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

- [20] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.
- [21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.
- [22] G. Marani, S. K. Choi, and J. Yuh, "Underwater autonomous manipulation for intervention missions auvs," *Ocean Engineering*, vol. 36, no. 1, pp. 15 – 23, 2009, autonomous Underwater Vehicles.
- [23] P. J. Sanz, P. Ridaou, G. Oliver, G. Casalino, Y. Petillot, C. Silvestre, C. Melchiorri, and A. Turetta, "Trident an european project targeted to increase the autonomy levels for underwater intervention missions," in *2013 OCEANS - San Diego*, Sep. 2013, pp. 1–10.
- [24] E. Simetti, F. Wanderlingh, S. Torelli, M. Bibuli, A. Odetti, G. Bruzzone, D. L. Rizzini, J. Aleotti, G. Palli, L. Moriello, and U. Scarcia, "Autonomous underwater intervention: Experimental results of the maris project," *IEEE Journal of Oceanic Engineering*, vol. 43, no. 3, pp. 620–639, July 2018.
- [25] A. Huster and S. M. Rock, "Relative Position Estimation for Intervention-Capable AUVs by Fusing Vision and Inertial Measurements," in *Proceedings of the 12th International Symposium on Unmanned Untethered Submersible Technology*, Autonomous Undersea Systems Institute. Durham, NH: Autonomous Undersea Systems Institute, August 2001. [Online]. Available: <http://www.stanford.edu/group/arl/cgi-bin/drupal/sites/default/files/public/publications/HusterR2001a.pdf>
- [26] E. Simetti, F. Wanderlingh, S. Torelli, M. Bibuli, A. Odetti, G. Bruzzone, D. L. Rizzini, J. Aleotti, G. Palli, L. Moriello, and U. Scarcia, "Autonomous underwater intervention: Experimental results of the maris project," *IEEE Journal of Oceanic Engineering*, vol. 43, no. 3, pp. 620–639, 2018.
- [27] F. Bonin-Font, G. Oliver, S. Wirth, M. Massot, P. Lluís Negre, and J.-P. Beltran, "Visual sensing for autonomous underwater exploration and intervention tasks," *Ocean Engineering*, vol. 93, pp. 25–44, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0029801814004090>
- [28] R. Dario Lodi, K. Fabjan, O. Fabio, and C. Stefano, "Investigation of vision-based underwater object detection with multiple datasets," *International Journal of Advanced Robotic Systems*, vol. 12, no. 6, p. 77, 2015.
- [29] A. B. Labao and P. C. Naval, "Cascaded deep network systems with linked ensemble components for underwater fish detection in the wild," *Ecological Informatics*, vol. 52, pp. 103–121, 2019.
- [30] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi, "Aggregating crowdsourced binary ratings," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13. New York, NY, USA: ACM, 2013, pp. 285–294.
- [31] D. R. Karger, S. Oh, and D. Shah, "Iterative learning for reliable crowdsourcing systems," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 1953–1961.
- [32] J. Xu and H.-S. Yoon, "Vision-based estimation of excavator manipulator pose for automated grading control," *Automation in Construction*, vol. 98, pp. 122 – 131, 2019.
- [33] B. O. A. Haugaløkken, E. K. Jørgensen, and I. Schjølberg, "Experimental validation of end-effector stabilization for underwater vehicle-manipulator systems in subsea operations," *Robotics and Autonomous Systems*, vol. 109, pp. 1 – 12, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921889018300952>
- [34] G. Antonelli, *Underwater Robots*. Springer, 2014.
- [35] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice-Hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.
- [36] "Bluerov2 heavy configuration retrofit kit." [Online]. Available: <https://bluerobotics.com/store/rov/bluerov2-upgrade-kits/brov2-heavy-retrofit-r1-rp/>

