UNDERWATER GESTURE-BASED HUMAN-TO-ROBOT COMMUNICATION

ROBERT CODD-DOWNEY

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

DECEMBER 7, 2024

Abstract

Underwater human to robot interaction presents significant challenges due to the harsh environment, including reduced visibility from suspended particulate matter and high attenuation of light and electromagnetic waves generally. Divers have developed an application-specific gesture language that has proven effective for diver-to-diver communication underwater. Given the wide acceptance of this language for underwater communication, it would seem an appropriate mechanism for diver to robot communication as well.

Effective gesture recognition systems must address several challenges. Designing a gesture language involves balancing expressiveness and system complexity. Detection techniques range from traditional computer vision methods, suitable for small gesture sets, to neural networks for larger sets requiring extensive training data. Accurate gesture detection must handle noise and distinguish between repeated gestures and single gestures held for longer durations. Reliable communication also necessitates a feedback mechanism to allow users to correct miscommunications. Such systems must also deal with the need to recognize individual gesture tokens and their sequences, a problem that is hampered by the lack of large-scale labelled datasets of individual tokens and gesture sequences. Here these problems are addressed through weakly supervised learning and a sim2real approach that reduces by several orders of magnitude the effort required in obtaining the necessary labelled dataset.

This work addresses this communication task by (i) developing a traditional diver and diver part recognition system (SCUBANetV1+), (ii) using this recognition within a weak supervision approach to train SCUBANetV2, a diver hand gesture recognition system, (iii) SCUBANetV2 recognizes individual gestures, and provides input to the Sim2Real trained SCUBALang LSTM network which translates temporal gesture sequences into phrases. This

neural network pipeline effectively recognizes diver hand gestures in video data, demonstrating success in structured sequences. Each of the individual network components are evaluated independently, and the entire pipeline evaluated formally using imagery obtained in both the open ocean and in pool environments.

As a final evaluation, the resulting system is deployed within a feedback structure and evaluated using a custom unmanned unwatered vehicle. Although this work concentrates on underwater gesture-based communication, the technology and learning process introduced here can be deployed in other environments for which application-specific gesture languages exist.

Acknowledgements

I would like to express my sincere gratitude to the exceptional team of commercial divers—Michael Jenkin, Heather Jenkin, James Zacher, and Philip "Bucky" Lanthier—whose invaluable contributions were essential throughout the research and development of this dissertation. Without their expertise, dedication, and tireless support, this work would not have been possible.

I also wish to extend my deepest thanks to my parents, whose unwavering encouragement and thoughtful guidance have supported me at every stage of my academic journey. Their belief in me has been a constant source of inspiration.

Lastly, but certainly not least, I owe a heartfelt thanks to my incredible wife, Meghan, and my son, Theodore, whose love, patience, and motivation helped me push through to the finish line. You both have been my rock, and your support has made all the difference during the final stretch of this endeavor.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

My thesis investigates the use of deep learning techniques to support gesture-based human to robot communication in the presence of the computational restrictions associated with computation performed on board an autonomous robot. In my thesis I concentrate on the use of the sign language used by SCUBA divers to explore and develop techniques to deploy human-to-robot communication infrastructure underwater, although the approach has broad applications in the terrestrial domain. The underwater environment places limits on computational resources (e.g., CPU + GPU) which are not necessarily constrained in terrestrial applications. It is thus almost an ideal application domain for this task. If it works here, it should work everywhere.

Clear, precise and well-defined communication is a necessary component of human-robot and human-machine interaction. Communication failures occur when there is a lack of communication or misunderstanding between parties. Failures as a result of either situation can range from merely frustrating to catastrophic. For example, a browser that neglects to display the size and progress of a file being downloading can be frustrating to the user. While a robot on a factory floor that does not communicate to surrounding workers its movement

envelope can be fatal. See [14] as an unfortunate, but real world, example of this later problem. A home assistant that doesn't understand a command to turn a light on can leave a user in the dark and frustrated. An artificial intelligence that misunderstands the intent behind a series of rules can be catastrophic. See [2] for a fictional example of this. While this last example is fictional, Isaac Asimov's "I, Robot" is perhaps one of first works on perverse instantiation in robotics[15].

Gestures are an essential form of communication for humans. Infants use deitic gestures (pointing) for the purposes of knowledge acquisition and establishing joint visual attention[69]. These type of gestures are common despite cultural differences. In later stages of cognitive development children begin to express both iconic and conventional gestures[107]. Iconic gestures visually reflect the action, object or attribute they portray, whereas conventional gestures can be culture specific such as nodding "Yes" or waving "Hello". There is even a growing theory that gestural languages pre-date the spoken word in human evolution[33].

Communication between human and robots is moving from keyboard inputs by machine specialists to multi-modal input from domain specialists. Yet many advanced robots show little improvement in terms of their human-robot interaction (HRI) or control from much earlier approaches. For example, the control of Tesla's remote controlled boat[76] demonstrated in 1898 exhibits many similarities to the tele-operational control of quad-copters today, where operators only have access to basic low-level functionality of certain actuators of the machine. However, this is perhaps not the best choice for Human-Robot Interaction (HRI). More sophisticated communication is facilitated via higher levels of autonomy. Consider, for example, NASA's Sojourner robot command workstation. This system used a graphical user interface and 3D display to enable operators to designate waypoints as a medium to communicate between the user and the robot[125][118].

As robots move into collaborative work environments with humans there is an increased need for the robot to be capable of interacting using the same modalities that human-colleagues already use to communicate with each other in that domain. One interaction technology that can be found in many human-human communication domains is the use of gestures; from babies communicating with their parents to disabled people communicating with their staff, gestures are a critical communication medium. It is to be expected that for appropriate domains robots should operate utilizing this modality as well. How can we enable such gesture recognition in human-robot/human-machine communication?

1.1 Gestural Communication

Gesture-based or "Sign" languages are a type of natural language that arises from the necessity to communicate without speech. This type of language is expressed as a series of manual articulations and sometimes non-manual expressive body language. Due to the visual nature of signing, only those currently viewing the signer receive the information being conveyed. In some cases this can be detrimental, especially when the intended recipient is not looking at the signer directly. Although in other cases this can be a benefit as it is possible to limit the recipients to those in the field of view. We can separate sign languages into two distinct categories; general purpose languages that support conversational interaction and languages developed for a specific applications or purpose.

1.1.1 General Purpose Gestural Languages

General purpose gestural languages are commonly found in hearing impaired communities and those who are physically incapable of using speech to communicate ideas and information[13].

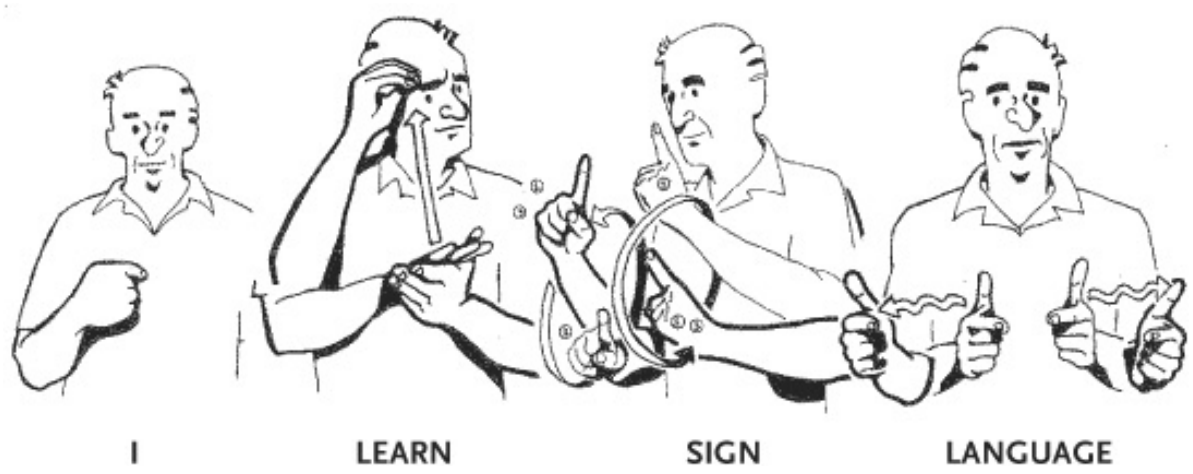


Figure 1.1: **American Sign Language**
American Sign Language gestures that represent the sentence "I Learn Sign Language". Reprinted from [3].

Perhaps the most well known of these if the American Sign Language (ASL). The US and Canada officially recognize ASL as a language. This has enabled ASL to proliferate throughout North America. However, ASL is not the only general purpose sign language. In other parts of the world a number of different sign languages have been developed. Some are based upon popular languages such as ASL while others have been developed locally within small communities. See [66] for a survey of the various sign languages used throughout the world.

A common first introduction to sign language is the alphabet. This often creates a misnomer about the complexity of the language as some assume the language is merely an exercise in spelling out words. In contrast, general purpose sign languages are very complex and capable of conveying complex ideas through a vast array of gestures. Consider, for example, Figure 1.1 which details four gestures which when shown in sequence represent the phrase "I learn sign language". Some may view this as grammatically incorrect and ambiguous, either representing the phrase "I'm learning sign language" or "I learned sign language". ASL lacks some common articles (a, an, the) and "to be" verbs (am, is, are,

(a) (b) (c)

Figure 1.2: **Application Gesture-based Languages**
A selection of images depicting professionals performing hand gestures related to their occupation. (a) Soldier giving the hand signal for "30 seconds". (b) A SCUBA diver giving the signal for "ok", (c) An aircraft marshal giving the signal for "STOP".

was, were), Grammatical structure is represented instead by word order[64]. This has the benefit of reducing the required vocabulary, as the sign for "learn" can be interpreted as "learn", "learning" or "learned", as appropriate. Emotion is conveyed by varying the speed and assertiveness of gestures. Facial expressions are also a key aspect of the language; a commonality between signed and spoken language. Facial cues in sign language are often over-expressed for clarity. The subtlety and expressiveness of general purpose sign languages such as ASL has frustrated efforts to develop machine algorithms that can effectively "read" ASL (see [109] for example) as most efforts focus on recognizing static gestures.

1.1.2 Application-Specific Gestural Languages

In contrast to general purpose gestural languages a number of special purpose or application-specific sign languages have been developed to deal with environments or tasks within which spoken languages are inappropriate. There are many examples of such application-specific gesture languages. Figure 1.2 shows examples drawn from the military, SCUBA and aircraft marshalling domains. The development of application-specific gesture languages is driven by the demands of the task. Typically, in these tasks a spoken language is not ideal. For

(a) Hold (b) Take Cover (c) Disperse

Figure 1.3: **US Army Hand Signals**
Various hand signals outlined in the US army handbook on visual signals. Reprinted from [120].

example, in some domains ambient noise can drown out the speaker even in close quarters. While in others, breaking silence can reveal one's location. Yet in others, natural language can be ambiguous depending on one's choice of words. An application-specific gesture-based language is a good substitute under these condition as gestures can be made silently and can be seen from moderate distances and under poor weather/lighting conditions. Due to the clear advantage that sign languages can provide in certain circumstances, a number of organizations/professions have developed and standardized their own sign languages that can be use to communicate common information needed within their field. A few of them are reviewed below.

Military Visual Signalling

The U.S. Military and other military, security and police organizations across the globe require mechanisms to communicate amongst themselves in close quarters without drawing attention to their location or message. Further complicating this process, these calamitous environments often have a large amount of ambient noise from the surroundings due to the

(a) Okay (b) Hold (c) Surface

Figure 1.4: **SCUBA Hand Signs**
Illustrations of hand gestures common to both recreational SCUBA divers and technical/commercial divers outlined by the Professional Association of Diving Instructors (PADI). Reprinted from [93].

perils of combat. Furthermore, security concerns may require the need for absolute silence for stealth purposes. Hand signals represent an inexpensive and easy solution to these problems. Hand gestures within this domain are most often used to relay instructions from a group commander/leader to subordinates (e.g., Hold, Fire, Echelon Right). Other gestures are used to relay information back to members of the group, often from an individual with a better vantage point (e.g., Enemy, Sniper, Gas)[120]. Examples are shown in Figure 1.3.

SCUBA Signs

The self contained underwater breathing apparatus (SCUBA) used by modern underwater divers typically inhibits the use of speech because the regulator is held within the mandibles using a mouth piece. Recreational SCUBA divers have two primary means of communication: written communication using slate and pencil are used to record information and communicate complex ideas, and hand gestures which are more commonly used to facilitate simple communication. Light and rope pull signals are also used in specific circumstances, particularly in commercial diving[93].

Recreational Diving. In recreational SCUBA diving the majority of hand signals are related to inquiring about the safety status of a diver, because this is the primary concern. Other signals are used to direct fellow divers to a goal. There also exist a number of gestures to notify others of the presence of marine wildlife (e.g., sharks, turtles, rays, etc.), although these gestures vary with geography and water conditions. This gestural language is simple and intelligible enough that most SCUBA divers can communicate with each other after a basic lesson. Figure 1.4 shows a few simple hand gestures commonly used by divers.

Technical and Commercial Diving. Technical and commercial divers have developed their own simplified set of hand gestures that are used to convey information necessary to their particular domain. There are a number of commercial and technical diving handbooks that outline numerous gestures to deal with specific domain problems such as: lines, entanglement, knives, and physical space. Unfortunately, the set of gestures used is sometimes contradictory as there are many organizations with no common international standard[29][93]. That being said, in a given geographical region a common set of gestures is used, and there is considerable overlap between recreational and commercial hand gestures for divers.

1.1.3 Overview

Application-specific hand gesture languages have been created for a variety of reasons, but mainly to suit a need within the domain. Despite their differences these gesture languages share many commonalities. They are highly structured, so slight variations in signings do not change the meaning of a gesture. They also typically obey a simple grammar that is intended to facilitate clear communication between parties. As a consequence of the intent for these gestures to be easily interpreted, it should be more straightforward to develop systems that

can understand such applied gesture languages rather than general purpose languages. Given this, and the need to provide effective human $\rightarrow$ robot communication in such domains. Here I consider applied gesture-based language understanding, In particular I consider the task in the context of robot assisted diving.

1.2 The Problem Domain

Commercial divers undergo a tremendous amount of cognitive loading while working in underwater environments. Typical tasks performed at depth include repairing, inspecting and building infrastructure in the hazardous underwater environment. Conventionally, aid in these assignments can be provided by deploying additional commercial divers that assist the initial diver with the task at hand. This practice can quickly become very costly, especially if a diver only requires a small amount of aid in accomplishing a task.

While current technology requires humans for complex manipulation tasks such as assembly of pipe fittings, robots are well suited for monitoring, recording and simple transport tasks in this domain. Critically, robots are immune to the effects of decompression sickness. This enables them to transport items to and from the surface and to divers operating at different depths. A robot's ability to record vast amounts of data allows them to be an effective means for task monitoring and evaluating a diver's work after completion. Robots are also capable of venturing into environments that may pose an unacceptable risk for a human diver.

Divers have developed a hand gesture-based communication language for underwater communication. This language relies on interpreting a sequence of hand gestures and is well understood by divers but is not well understood my machines. Hand gesture-based communication is not widely employed in the domain of underwater robotics, although there have been some limited efforts in this direction, e.g. [47]. This form of communication

Figure 1.5: **Commercial Divers at Depth**
A pair of commercial divers welding an underwater pipe.

is however commonly used among both commercial and recreational divers and this type of communication is preferable to other methods currently in use for human to robot communication underwater such as tag libraries and bulky tethered communication devices. A brief review of human-robot underwater communication approaches can be found in Chapter 2.

1.3 The Basic Approach to Gesture Understanding

Supervised CNN's based on transfer learning have proven effective in capturing and categorizing individual hand gestures, and this, coupled with network structures that integrate temporal cues (e.g., RNN's and especially LSTM's), would seem to be an appropriate structure for recognizing communications given in an application-specific gesture language. A

key problem in such an approach is the complexity of obtaining, annotating and curating a sufficiently large training dataset for both the specific gestures used as well as the likely expressions that will be communicated from the user to the robot. Rather than capturing and labelling such data explicitlty, my thesis develops a weakly-supervised technique to reduce the burden of annotating large amounts of images and the organizational structure of annotations is also leveraged to create synthetic data sequences to support Sim2Real learning. This data is used to train an LSTM-based system to translate diver gesture sequences into robot command sequences. The effectiveness of the approach is demonstrated through laboratory evaluation using data captured underwater in swimming pools and in the open ocean. Further on-line demonstrations of the approach affords the ability for the gesture recognition system to interact with the user, enabling feedback from the recognition pipeline to improve communication performance

The system developed in my thesis provides gesture-based human->robot communication using an application-specific gesture language, specifically SCUBA gestures. Although much of the training and analysis takes place in an offline manner, the intent is to integrate a weakly supervised neural network architecture trained on a Sim2Real dataset that processes video sequences within a feedback structure to (i) provide concrete feedback to the user that individual gestures have been detected by the robot and (ii) to provide a final confirmation of the command sent to the robot prior to its execution. An example of this feedback system and the system developed in my thesis is shown in Figure 1.6.

1.4 Contributions

This document outlines the method for the development and evaluation of application-specific gesture language recognition (PADI SCUBA Signals). This work:

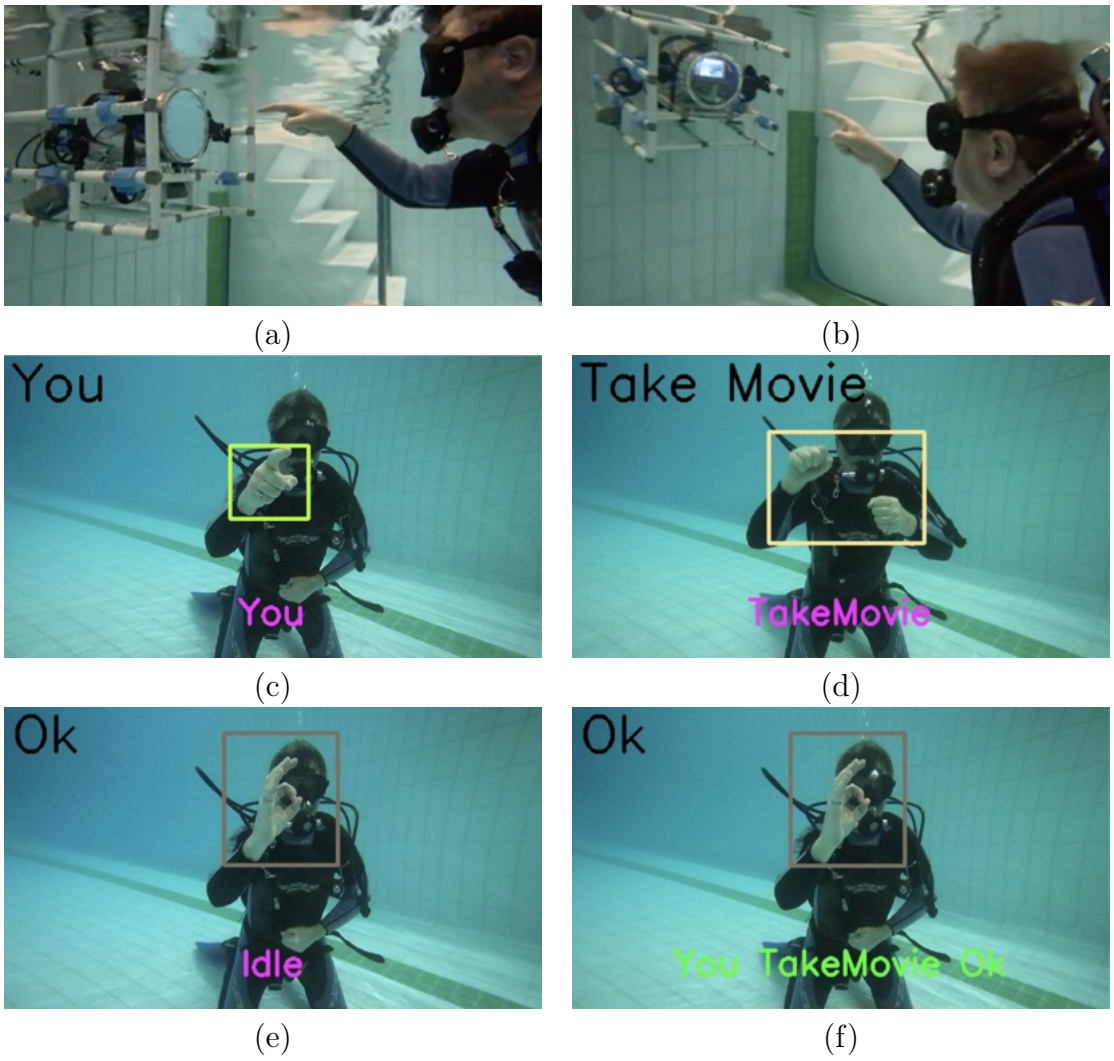


Figure 1.6: **Gesture Detection and Feedback**
Sample interaction of a diver with the robot while the diver signals YOU TAKEMOVIE OK. (a)-(b) show the diver interacting with the robot. The robot monitors the interaction through a forward facing camera and shows the diver the state of the interaction through a display mounted at the front of the robot. This display is visible in (a) and (b). (c)-(f) show the display on the robot presented to the diver. The background image is the camera view with a bounding box showing the instantaneous gesture recovered in the upper left of the image. The most likely token in the communication sequence is shown in purple underneath, and when a valid communication sequence is recovered this is shown in green. Once a full sequence is recovered the diver can signal acceptance of this using the sequence YOU OK.

- Develops the SCUBANetV1+ dataset, a hand labelled dataset of diver parts. This work is described in [20]. SCUBANetV1+ can identify SCUBA divers at depth and

their primary body parts (head, body, hands, etc.).

- Develops and leverages a general framework for *weakly-supervised learning* to learn both the individual gestures as well as the likely expressions to be communicated without the tedious, error-prone, and time consuming process of labelling such data. This approach reduces by several orders of magnitude the labelling work that would be required for a fully supervised approach to training the model. This work is described in [21].

- Develops a weakly labelled dataset of diver gestures. This dataset contains over 93,000 weakly-labelled diver images, and can recognize 30 diver gesture classes.

- Develops a sim2real approach for constructing SCUBA gesture sequences. This partially-simulated dataset contains over 89,000 gesture sequences drawn from 29 different gesture classes. The trained network SCUBAGesture is evaluated through both lab-based evaluation of datasets as well as through controlled experiments with divers.

- Embeds this gesture recognition process within an interactive mechanism that provides feedback to the user when deploying gesture recognition algorithms in real world scenarios.

- Demonstrates that the resulting system can be used to communicate with an ROV that was developed within this thesis in a pool setting, operating within the computational and sensing limits of the device.

Fundamentally, the process of communicating with the robot involves capturing hand gestures in the sequence of frames presented to a robot and then characterizing these frames into sentences. Here I provide a high level view of the architecture and the key technical aspects associated with the architecture.

Figure 1.6 summarizes the experience from the divers' perspective. At depth the diver communicates with the robot using hand gestures. A forward facing camera in the robot captures the diver and their gestures and also provides visual feedback to the diver to support robot->diver communication Figure 1.6(a)-(b) shows the external view of the process. As the diver interacts with the robot the recognition system provides the diver with feedback as to what gesture the robot is recognizing. In essence, this provides both instantaneous frame by frame labelling of the diver's gestures as well as the state of the entire communication. Figure 1.6(c)-(f) provides snapshots of this interaction. Commands from the diver to the robot are book-ended with 'YOU' and 'OK' signals. Each individual gesture in the sequence is echoed visually to the diver, and once the 'OK' bookend is received, the full command is displayed. An acknowledgment is then sent by the diver (YOU OK sequence), and the command executed by the robot.

Each frame of the video captured by the robot is processed to extract hand gestures found in the standard set of SCUBA hand gestures. The process of extracting these gestures involves training a neural network to recognize the set of gestures and the head of the diver. Transfer learning is used to train the network. Rather than hand labelling the 100,000+ necessary images for this process, a weakly supervised process is used to leverage a much smaller manually labelled dataset and structured data collection. Details of the hand labelled dataset process (ScubanetV1+) is given in Chapter 4. The process of using this in a weakly-supervised fashion to label frames with specific hand gestures is described in Chapter 5.

Once the frames have been labelled, a seq2seq process is used to map gesture sentences to "English language" sentences. This process uses a LSTM-based temporal network to represent mappings from gesture sequences to sentences. Again, rather than manually labelling 100,000+ such sequences we utilize a weakly-supervised process to generate synthetic sentences from

real data sequences to train the network. This helps to cross the sim2real gap. This process is detailed in Chapter 6.

The algorithm described in Chapter 6 is described in terms of an offline process that takes token sequences and develops a textual description. This process is then integrated into an interactive system and tested by a commercial diver. Results of this test are provided in Chapter 7. Details of the robot developed for this project can be found in Chapter 3.

Chapter 2

Background

Gesture based communication is a popular theme in the human computer interaction (HCI) and human robot interaction (HRI) communities. There exists a number of gesture-based systems used for a range of applications described in the literature. For example: Hand waving gestures have been used to capture the attention of a drone from far away and take a picture[73][74]. Pointing gestures have been used to navigate within virtual environments[23]. Motion based gesture systems like the Playstation Move[91][90] have been used to control the armature of avatars within a 3d environment using special controllers for more interactive gaming experiences. A number of technical approaches have been developed to enable these interaction systems. Some of these are discussed below. We begin, however, with a survey of gesture-based robot communication systems that have been developed. We then move on to review technologies that underlie these approaches.

2.1 HRI using gestures

There exists a large and growing literature describing mechanisms that utilize gestures to control autonomous systems. Systems have been developed for a wide range of different robot deployments, including underwater, in the air and on the ground. One clear distinction between the various techniques are those that rely on augmenting the human with some worn monitoring technology or some special sensing technology, and those that rely on passive sensors such as vision. Regardless of the measurement technology used, in many cases the problem is formulated as one of mapping signals to some, often task-specific, word or command set. Gestures can be based on configurations of the body or the hands, or the display of specific codes through a range of different technologies. [9] provides a recent review of a different approaches to the problem.

As the environment often dictates the types of technology that can be applied to the problem, and indeed the severity of the problem itself, this review is organized by the area of application.

2.1.1 Underwater

The underwater environment is complex in that it prohibits a range of different technologies (e.g., RF-based techniques) and limits the applicability of worn technology. As a consequence vision-based approaches are common, with older approaches relying on traditional image-processing techniques while more modern systems typically leveraging AI-based systems. Given the prevalence of a standard underwater gesture language, this is often used as a basic language set although some approaches suggest a novel set of gestures.

RoboChatGest

RoboChatGest[47] is a diver to robot gesture language used to dynamically reconfigure an underwater robot's mission parameters during operation. The system uses a neural network to recognized pairwise combinations one of 10 one-handed gestures that are presented simultaneously. Gesture pairs map one-to-one to a set of commands which can be strung together to define a complex operation.

While the 10 gestures adopted by the system are familiar to SCUBA divers, there meaning is distorted within the context of the gesture language which could be a source of confusion. Additionally the gesture pair mappings are hardly intuitive which could lead to further frustration.

Caddian

Caddian[18] is another gesture language for diver to robot communication. Caddian builds upon traditional SCUBA gestures by defining a formal grammar and includes additional gestures to help structure communication. Gesture detection is accomplished using traditional computer vision techniques aided by special gloves augmented with visual markers. Due to this gestures in the language need to be carefully selected.

2.1.2 Aerial

Aerial environments provide vast swaths of three dimensional space where UAVs are subject to various environmental conditions (e.g., wind, fog, rain). As these robots must overcome gravity by generating their own lift their payload capacity often limits onboard computational and sensing capabilities. Generally operating outside the typical range of WIFI and bluetooth signals, other RF-based technologies are often used in this domain for control and monitoring.

There have been efforts to use vision sensors to detected gestures for the purpose of controlling these types of robots. Gestures in this domain need to be highly visible at long ranges and tend to be overly exaggerated to draw attention. Bruce et al. [73] utilize double arm waving gestures to establish attention between robot and operator. Pfeil et al. [88] explore the use of arm movements in various metaphorical control systems (e.g., pointing, joysticks). UAV-Gesture[87] used a subset of aircraft and helicopter handling signals for basic navigational control. Lee and Yu [61] used a wearable device with inertial sensors and tactile feedback that could sense hand motions which mapped to basic flight controls (e.g, up, down, left, right, forward, back, stop).

2.1.3 Outdoor

Outdoor terrestrial environments vary widely in their composition and structure while also being subject to both adverse and favourable weather conditions. Robots operating in this domain have few limits on the communication technologies available to them. Often tablet and joystick devices can be utilized easily and effectively rendering gesture based control a seldom used alternative. However some scenarios preclude the use of additional equipment for command and control. Collier et al. [17] presents a system to control a robot using military hand signals detected using a high density lidar scanner.

2.1.4 Summary

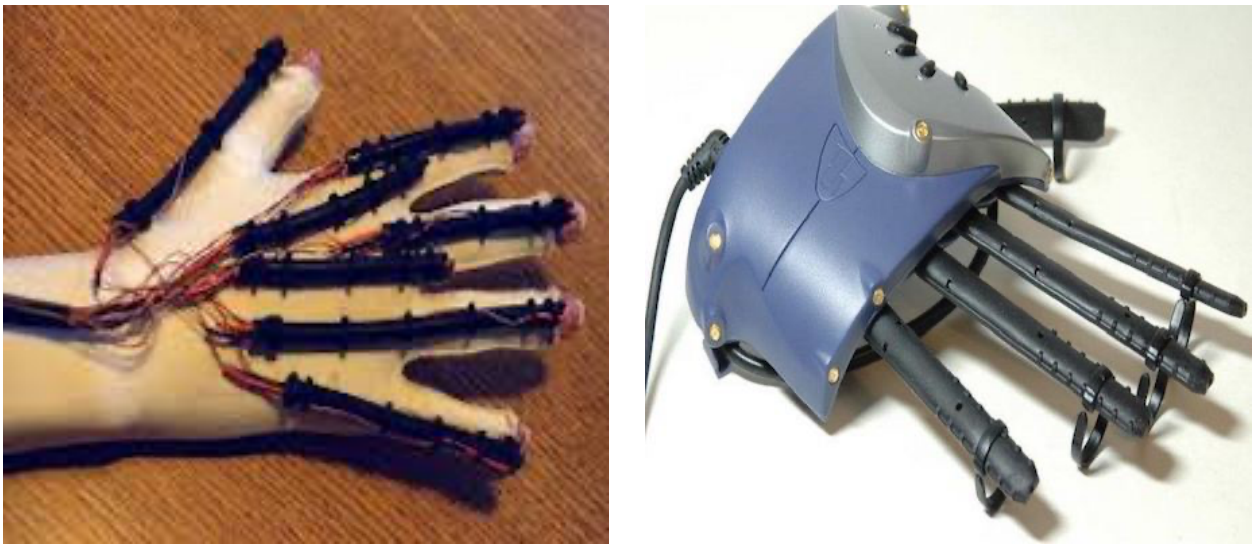
Gesture-based human->robot communication is complicated by (i) the need for a communications strategy that works with the reality of the operational environment and (ii) the need to choose an appropriate gesture language and mechanism. In terms of (i), some gestures are going to be more appropriate than others for a given domain. For the aerial domain,

for example, the potentially large distance between the UAV and the human requires that the gestures be of sufficient scale to be viewable. In contrast to the underwater domain, the nature of the underwater environment precludes long range communication via gestures. Turbidity in the water column will likely obscure gestures long before relative scale becomes an issue. In terms of (ii), the development of a custom communication language may simplify some tasks, such as reducing the communication set to a very manageable set of symbols, but at the same time it requires considerably more effort in terms of training the human operator and complicates the process of finding humans that are suited to labelling datasets for machine training. Artificial augmentation of the user to enhance the communication strategy may be appropriate for some domains, but unless the augmentation is already part of the task, this will increase operational cost and the cognitive load of the human.

It is also important to recognize that the specific message being communicated is part of a larger conversation. It is thus important to embed the operator->robot message within this larger scale interaction. A number of different approaches are possible here, beyond just observing the robot to confirm that the robot is doing what was intended. For example, [41] and [133] describe systems that augment the robot with a display that allows the human to understand exactly what the robot has internalized.

2.2 Augmented Techniques

Given the complexity of understanding hand or body configurations using passive technologies, augmenting the human so as to simplify this task is a common approach. Handheld or wearable technology designed to measure and sense the movement and/or state of the user and their hands can be used to simplify the sensing and inference process. Some of these methods outfit the user with a variety of sensors to directly measure the angle of each joint (e.g. [27]), while



(a) Sayre Glove (b) P5 Gaming Glove

Figure 2.1: **Data Gloves**
(a) the Sayre Glove was one of the earliest implementations to measure finger poses. (b) the P5 Gaming glove represents a design a few generation ahead of the Sayre Glove.

others use clothing with visual markers to detect points on the body (e.g. [72][58][19][127]). Handheld methods refer to techniques that use a controller (e.g. Figure 2.2) held in the user's hand.

Augmented techniques trade off the advantages of reducing or even eliminating the effort required to capture user intent at the cost of adding equipment to the user and between the user and the robot. The actual technology used for augmentation may also preclude its use in challenging environments, such as underwater.

2.2.1 Data Gloves

Early approaches to hand gesture recognition in the fields of human-computer interaction and human-robot interaction (HCI/HRI) explored wearable devices that measure the actual state of the joints of the hand (see Figure 2.1a). Data gloves like the Sayre Glove[27], augment

the user with gloves and multiple sensors to detect hand pose. While these approaches are accurate and effective the need for special equipment limits their application. The glove itself can be limiting and a further complication results from an often physical tether from the glove to the remote computational hardware. Such limitations are of particular concern in industrial applications where cumbersome equipment is a deterrent to free motion and productivity.

2.2.2 Markers

As more computing power became available and vision-based sensors became less expensive, approaches began to use cameras to detect hand gestures using a variety of vision based marker tracking methods. Perhaps the most well known of these systems is the Optitrack system[83]. This system uses a constellation of of retroreflective reflectors and spectral IR lighting to track body position. Other vision-based marker systems like ARTag[36] and AprilTag[81] use known 2-dimensional black and white bit matrices to help calculate the pose of the tag relative to the camera. These vision-based marker systems remove the need for users to wear custom made electronics to track movement or position but require that the user be augmented with these targets. While some of these properties can be beneficial, a major reason to use them for human robot interaction in an underwater environment is the relatively limited amount of processing power required to find them within an image and then to decode them. The use of a set of unique tags allows for the development of a simple command language, where each tag corresponds to a different command. Sequences of tags such as those used in RoboChat [31] can form a rudimentary language to describe tasks. This form of communication is effective but cumbersome, as operators need to carry a library of tags. Finding the specific tag needed for the next part of a command sequence is arduous

(a) Playstation Move Controller (b) Playstation Eye

Figure 2.2: **Playstation Move**
Sony Playstation devices used to measure 3D movement in the real world and interact with virtual reality. (a) a Playstation Move Controller contains inertial sensors to measure orientation and coloured visual marker. (b) the Playstation Eye Camera is used to track the location of each coloured markers in the scene.

and it is possible to accidentally show the wrong card while searching for the correct one. The use of custom or dynamic tags has also been explored[119]. This technique utilizes a tablet-like device that allows the user to select a series of control commands. Once verified a compact sequence of tags that encode the command sequence is generated and shown to the robot.

2.2.3 Mechanical Controllers

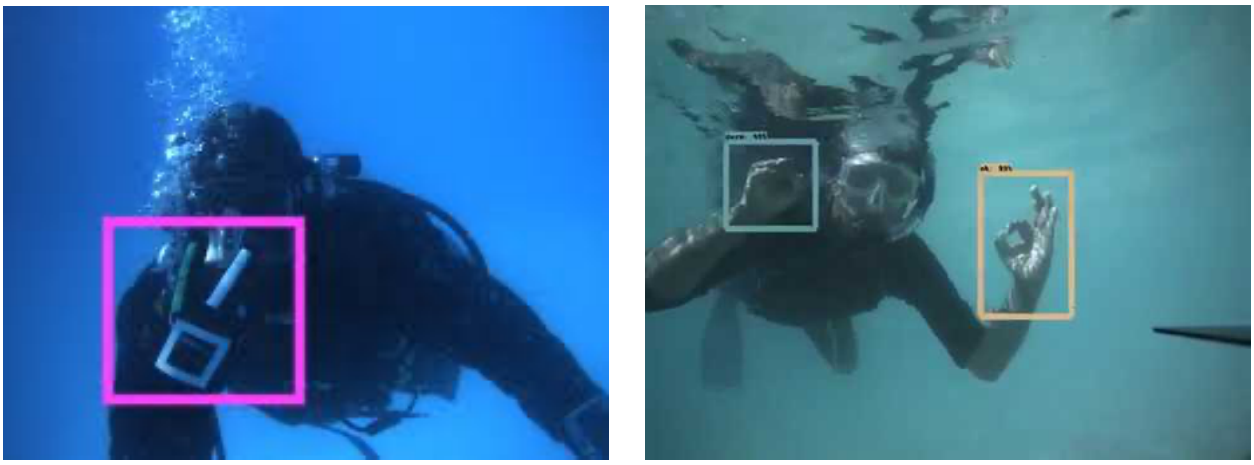
Perhaps the most well known technique for capturing gestures is through the use of hand held controllers, popularized by commercial products such as the Nintendo Wii[123] and PlayStation Move[91] (see Figure 2.2). These controllers house a variety of inertial sensors to detect their orientation in space and transmit this information back to the console via a wired or wireless (e.g, Bluetooth) connection. Positional tracking is accomplished using a visual

marker at a known reference point, like the Wii's sensor bar[124] or tracking a visual marker of a known size using a camera such as the Playstation Eye[90] (see Figure 2.2b). Controllers have the benefit of being easy to pick up and let go. Unfortunately they also use up the user's hand, limiting their ability to contribute to solving other problems concurrently. Existing hand held controllers are also generally not designed to be used in challenging environments (e.g., outdoors) and the communication infrastructure required to pass information from the controllers can be prohibitive in certain environments.

2.3 Uninstrumented Techniques

Uninstrumented gesture recognition removes the need to outfit the user with any special clothing or equipment affording them to move more freely. Generally such approaches utilize cameras to capture images of the hands or the body making gestures. Image sequences are used to understand the gesture being made. Given that hand and body gestures are typically three dimensional and regular cameras do not capture depth information such approaches must either recover this information or infer gestures without it. Capturing depth information can be accomplished using RGBD cameras or through stereo vision-based approaches[75].

Commercial RGBD cameras such as the now discontinued Microsoft Kinect[53] and the Leap Motion[59] extract three-dimensional geometry from the scene using stereo matching, improved by projecting an infrared pattern (structured or unstructured light) into the environment. This technology works well in terrestrial environments where infrared light is not highly attenuated. For example, Leap Motion has been utilized effectively as an input device for virtual reality in a variety of ways, see [23][132][113]. Unfortunately this technology is rendered less effective in the underwater domain where light is attenuated more than in air. Advances in general purpose graphics processing units (GPGPUs) have enabled stereo



(a) Coloured Markers (b) Static Hand Pose

Figure 2.3: **Hand Signals**
Current methods for detecting hand signals use either gloves with colour markers (a) or a combination of static hand poses (b) without visual markers. (a) is reprinted from the European Caddy project[72], (b) is reprint from [47].

cameras obtain results similar to RGBD cameras without the need for infrared light.

The use of hand signals is a commonly used method of diver-diver communication [6]. However, there are few examples of their use in the underwater domain for human-machine interaction, most likely due to technological constraints and a lack of publicly available data to feed into data driven machine learning approaches. One example of hand gesture-based human-robot interaction used underwater is the Caddy project [72]. This joint venture relied on a number of coloured markers on the user's glove in order to detect palm and finger placement (hand pose) to properly detect hand gestures. While this is a tremendous step forward, the reliance on predetermined visual markers on gloves prevents this work from being widely applicable. Another approach [47] uses a small set of easily recognizable hand poses, which when presented using both hands provides a large vocabulary that is available to the user. Unfortunately, learning to use this means of communication can take time, the proposed combination of gestures is not very intuitive, and the language does not map onto

the common gestures used SCUBA divers.

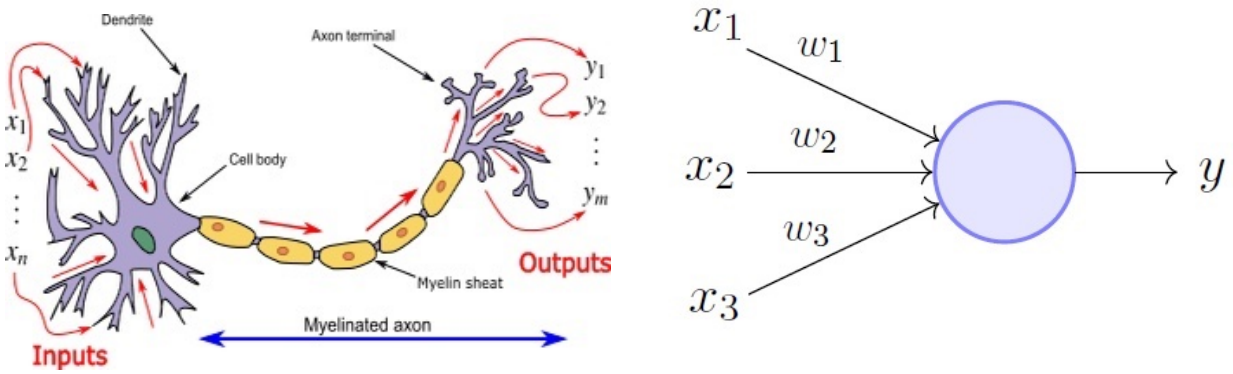
Given the difficulty of communication based on hand gestures, another approach is the use of whole body gestures as described in [55]. Although there exists a number of high performing body tracking systems (e.g., Yolo [49] and OpenPose[16]) such systems are not well trained on underwater imagery, nor do the gestures map onto the normally used SCUBA signals. As a further issue, actually performing large motions especially with flippers in the water column will cause the diver to move, which may, or may not, be a desirable outcome.

2.4 Neural Networks for Gesture Recognition

Given either a RGB or RGBD image sequence of a potential gesture it becomes necessary to extract the gesture and its intent. Prior to the mid 2010's algorithmic approaches on traditional image processing techniques were popular. More recent approaches typically rely on data-driven algorithms which achieve better results at the expense of the lack of an explainable deterministic method. Some of these approaches are reviewed further below. Artificial neural networks, neural networks for short, are a computational architecture that utilizes collections of artificial neurons that mimic the computational capabilities of a biological neurons (see Figure 2.4). Each neuron computes an output value based on a weighted sum of its input values and an activation function.

2.4.1 Perceptron

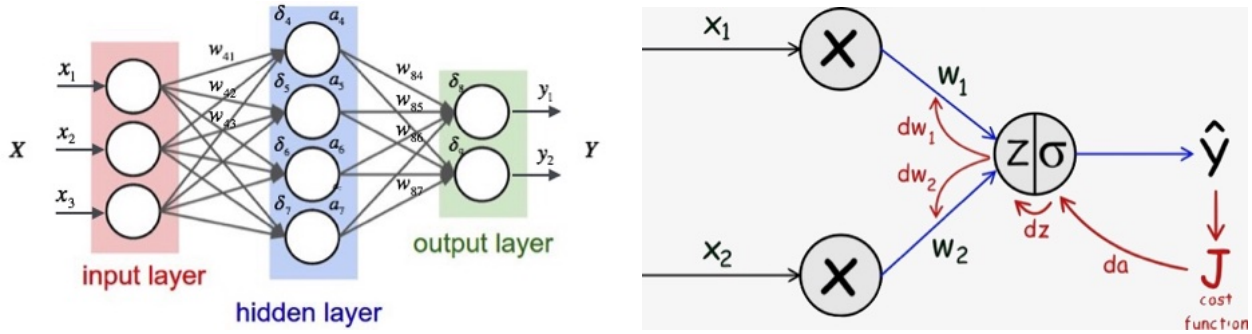
The perceptron and the perceptron algorithm[101] are the fundamental building blocks of many neural network architecture. The perceptron algorithm is a binary linear classification algorithm, capable of distinguishing between two input classes separated by a hyper-plane.



(a) Neuron (b) Perceptron

Figure 2.4: **Artificial vs Biological Neurons**
(a) an illustration of a neuron inside a human brain. Image reprinted from[8]. (b) an abstract representation of a digital neuron called a perceptron. Image reprinted from[104].

Originally conceived as the panacea to the artificially intelligence problem by the U.S Navy[80] the perceptron was thought to be capable of speech, locomotion, vision and self replication. Initially the single layer perceptron models showed great promise. However, it was quickly proved that these models were incapable of representing common simple patterns (e.g., the XOR function). This led to a huge decline in the field of neural networks for a number of years. It wasn't until the mid 1980's that the power of feed-forward networks and multilayered perceptrons eclipsed the shortfalls of the original algorithm using a training method known as back-propagation[60]. The back-propagation algorithm computes the gradient of a chosen error function at the output and adjusts the incoming weights to minimize the error. These results are then propagated to the previous layer of the network where the algorithm operates on each node in the layer. The structure of the perceptron was biologically-inspired by neurons in the human-brain. See Figure 2.4 for a visual comparison.



(a) Artificial Neural Network (b) Backpropagation

Figure 2.5: **Neural Network and the Back-propagation algorithm**
(a) an illustration of a simple artificial neural network with a single fully connected hidden layer (b) an illustration of the back-propagation algorithm used to update the parameters of nodes within and neural network.

2.4.2 Function Estimation

The goal of supervised neural networks in the field of machine learning is to accurately estimate the continuous nature of an unknown high-dimensional function given a set of known discrete inputs and their corresponding outputs. There are two major factors that contribute to how accurate the neural network can estimate the real world function. The first is the variety of data used to train the network, and how this variety reflects real world circumstances. The second is nature of the components (layers) and complexity of the neural network being trained. The first problem is often dealt with by acquiring more training and testing data. The second problem is often referred to as a model search, changing the configuration of the neural network in order to better estimate the hidden function using the provided dataset.

2.4.3 Image Classification

The image classification problem deals with assigning meaningful labels to an image to classify it and is considered to be a fundamental problem in the field of computer science. In 2005 the Pascal Visual Object Classes Challenge (PascalVOC[34]) launched a yearly competition to benchmark the performance of image classification algorithms. At its inception the challenge dataset contained 4 object classes with over 1,500 annotated images. When the challenge ended in 2012 it contained 20 objects classes with over 11,000 images with annotations that included region of interests (ROIs) for object detection algorithms and pixel segmentation for scene segmentation algorithms.

In 2010 the The ImageNet Large Scale Visual Recognition Challenge (ILSVRC [102]) launched another yearly competition that quickly became the primary benchmark for image classification and object recognition. The reason for this is because the ImageNet dataset contains over one thousand object classes with over one million annotated images. Over the years ImageNet has gradually increased the number of annotated images per category. The current version of ImageNet[46] has over 14 million images across 21 thousand different synsets/categories. Prior to the introduction of AlexNet[56] in 2012, the most advanced algorithms in the field employed a variety of classical computer vision techniques for object recognition. Since 2012 convolutional neural networks have been the primary tool through which approaches such as Inception[111], MobileNet[103], ResNet[43] and YOLO[98] have achieved tremendous success. The success of these algorithms have prompted numerous companies to begin integrating the technology into commercial products.

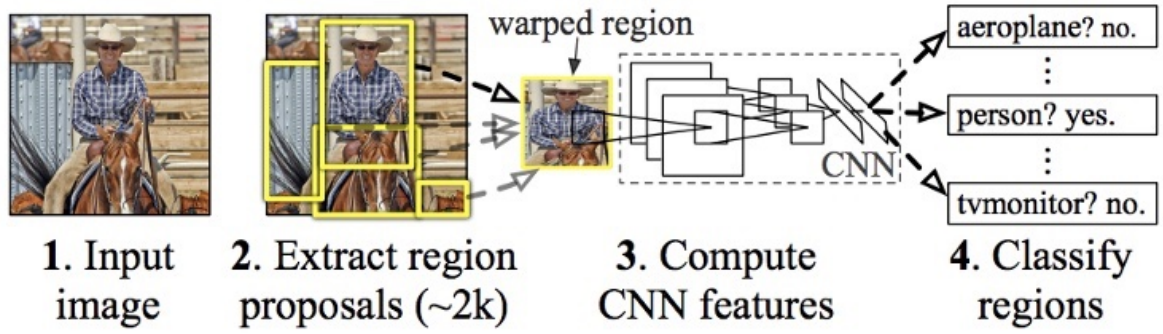


Figure 2.6: **R-CNN**
A diagram illustrating the major components of the R-CNN algorithm. The input images in split into 2 thousand regions each which are separately fed into a image classification CNN. Image reprinted from[40].

2.4.4 Object Detection

In computer vision the subfield of object detection is very closely related to the image classification problem and in many way can be viewed an extension of the field. This is because object detection can be broken down into classifying a number of subregions within a single image. This relationship has becomes more evident when reviewing recent approaches in the field. Research in Neural Networks for this problem have resulted in a number of architectures that have proven very effective. A number of the more successful architectures are reviewed below.

R-CNN

The R-CNN[40] algorithm (see Figure 2.6) was one of the first successful neural networks to address the problem of object detection. The first stage of the algorithm generates 2000 region proposals within an image. These regions are then fed separately to a pre-trained convolutional neural network (CNN) to extract image features as a 4096-dimensional feature

vector. This feature vector is then fed into a support vector machine (SVM[77]) to detect the presence of an object within each of the proposed region of interest (ROI). The SVM also produces four offset values to help adjust the size of the proposed ROI in cases where the object is partially cut-off.

Unfortunately this algorithm has significant drawbacks. The requirement to run a CNN model 2000 times introduces a tremendous computational requirement. Additionally, the regional proposal algorithm is fixed and thus cannot benefit from the model training process. With a run time on the order of fifty seconds per frame the R-CNN algorithm is generally unsuitable for real time operation.

Fast R-CNN

The Fast R-CNN[39] algorithm is an adaption of the first R-CNN algorithm published by the same authors that addresses a number of the shortcomings of the original R-CNN algorithm. Instead of feeding each of the 2000 region proposals into a CNN, only the input image is fed into a CNN to generate a single feature map. Using this feature map a number of ROIs are then generated using the selective search algorithm. Each ROI is then warped into a square and reshaped to a fixed size using an ROI pooling layer so that it can be fed into a fully connected layer producing a ROI feature vector. The ROI feature vector is then utilized by two soft-max regressors; the first predicts the object class, and the second produces offsets to increase the precision of the bounding box.

The Fast R-CNN algorithm is much faster than the R-CNN algorithm because it addresses R-CNN's primary bottleneck; running 2000 individual CNNs. The decision to produce only a single feature map using a CNN from the input image, and generating region proposals from this map reduces the computational complexity of the algorithm substantially. However, the

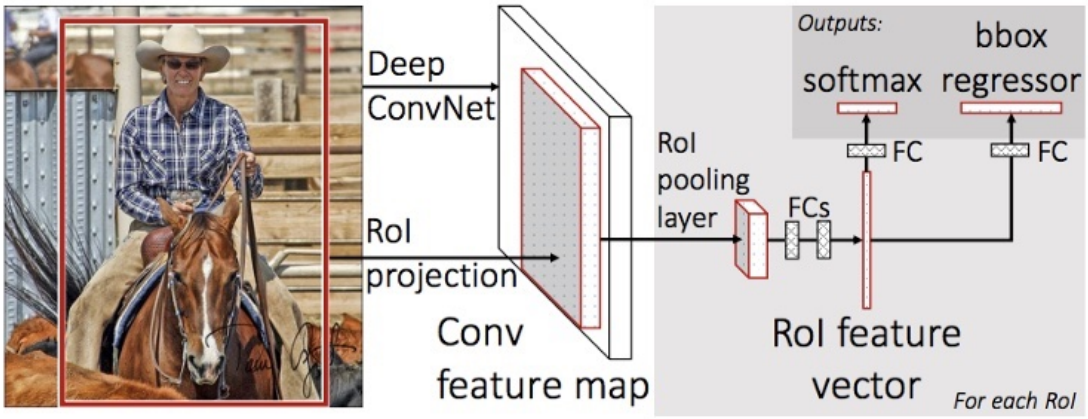


Figure 2.7: **Fast R-CNN**
A diagram illustrating the major components of the Fast R-CNN objection detection algorithm. Fast R-CNN generates a feature map using a CNN before generating its 2000 object proposals. This approach avoids repeat use of the CNN. Image reprinted from [39].

use of selective search and subsequent network used on each proposed ROI presents a new bottleneck.

Faster R-CNN

The Faster R-CNN[100] (see Figure 2.8) is an iterative improvement of the R-CNN algorithm. Unlike its previous revision (Fast R-CNN) Faster R-CNN is capable of real-time performance on high-end hardware. Both the R-CNN and Fast R-CNN use a selective search algorithm to generate region proposals at varying points within their processing pipeline. This selective search algorithm[117] is computationally intensive and cannot benefit from training like other parts of the algorithm. The Faster R-CNN algorithm uses a regional proposal network instead that is capable of learning to generate improved region proposals. The feature map and region proposals are then fed into a network for classification, the architecture of which is modelled tightly after the Fast R-CNN network that performs the same function, the only

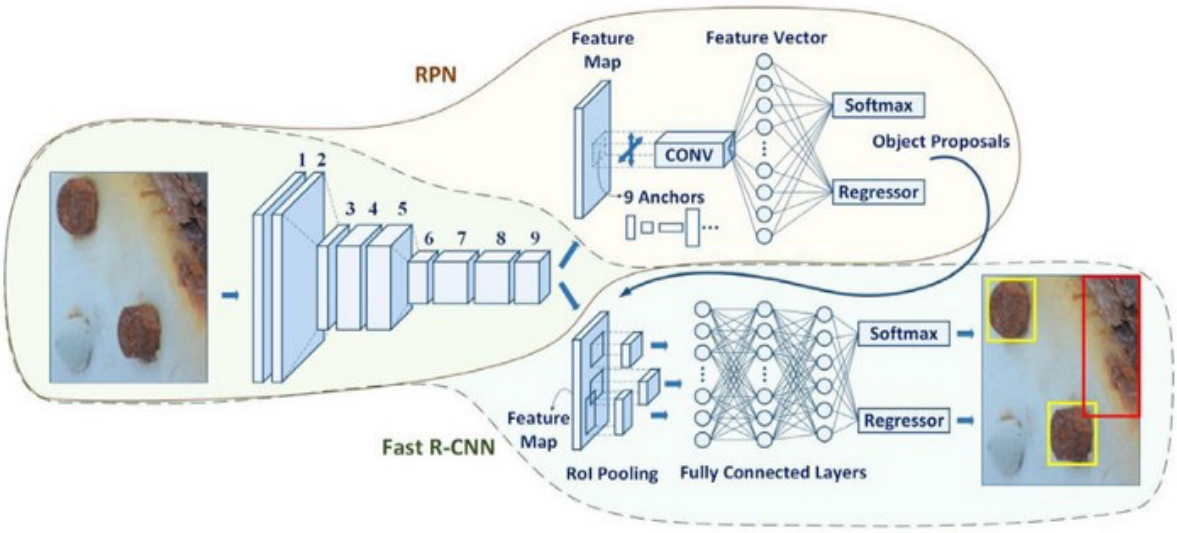


Figure 2.8: **Faster R-CNN**
A diagram illustrating the three parts of the Faster R-CNN design; feature extraction using the final pre-classifciation layer of a CNN, region proposal, object classification. Figure reprinted from [52].

difference is regions-of-interest are processed in tandem.

YOLO

The YOLO [98] object detection algorithm uses a much different strategy from the R-CNN approaches above. Instead of using pre-trained classifiers (see Figure 2.9) and region proposals to drive object detection, YOLO attempts to unify these components into a single neural network. The algorithm divides the input image into SxS grid cells (where S is an algorithmic parameter). The detection of an object is the responsibility of the grid cell in which the object's centre lies. Each grid cell predicts B bounding boxes with a corresponding confidence value. The confidence value represents the intersection over union (IOU) between the predicted bounding box and the ground truth. Each grid cell also computes a set of C conditional class probabilities. The class probability map is combined with the bounding boxes and confidence

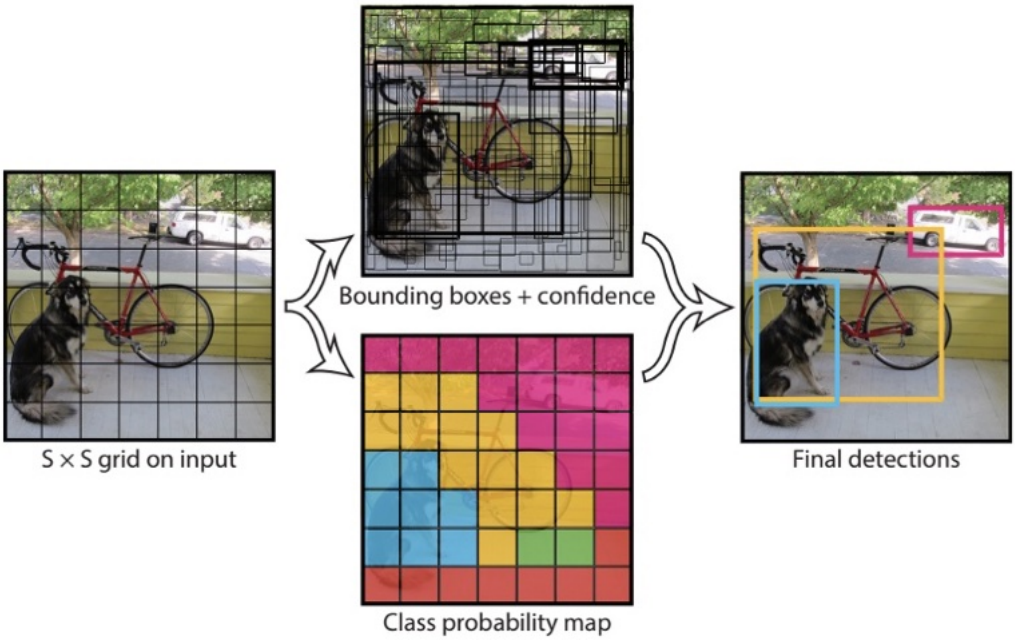


Figure 2.9: **YOLO Objection Detection**
Image reprinted from[98].

values to yield class-specific confidence scores for each bounding box and predicted IOU scores. These IOU scores reflect how well the predicted bounding box fits the detected object.

This neural network architecture is an order of magnitude faster than Faster R-CNN and can achieve similar accuracy on popular datasets. However, the algorithm is not without its drawbacks. The spatial constraints imposed by the SxS grid cells and B bounding boxes per cell prevent YOLO from detecting small groups of objects (e.g., a flock of birds). Increasing the discretization of the grid and predicting more bounding boxes can improve results at the expense of real-time performance.

YOLO has gone through a number of refinements since first introduced. This work uses YOLOV8[49] and is available in a range of different architecture sizes. YOLO is trained on the MSCOCO Dataset and is typically used as a base model upon which application-specific detection models are constructed. This is the approach that is followed here, as is described

Figure 2.10: **Objection Detection - Transfer Learning**
An image processed by a neural network that has be retrained using transfer learning on an underwater dataset SCUBANet[20].

in Chapters 4 and 5.

2.4.5 Object Detection – Transfer Learning

Object Detection neural networks are trained using benchmark datasets such as ImageNet[102], MSCOCO[65] or the Open Images dataset[57]. While these networks are capable of detecting a large number of object classes, sometimes the pre-defined set of objects do not conform well to the application domain of interest. In such cases a different set of objects are required to be detected within an image and a new dataset must be constructed in order to properly train the network. Fortunately a technique called transfer learning[67] can be used, this method trains a new classification layer while keeping the rest of the network frozen. We have demonstrated the effectiveness of this approach in recognizing the body parts of divers in multiple underwater environments. A hand labelled set of salient features of a diver (head,

body, flipper, hand) is collected. A standard object detection network is frozen except for the final layer, and this final layer is then retrained using the labelled data. A modified version of the original SCUBANET paper [20] is used in this work.

2.4.6 Improving Performance

A number of data augmentation techniques are employed to improve the accuracy of a deep-learning model on a particular dataset (see [128] for a review). The most common of these approaches involves duplicating images in the dataset and adding varying degrees of different kinds of noise whilst preserving the correct label. This can enlarge a dataset and helps prevent overfitting. Another common technique is to include background images that do not contain any labels, as this can help prevent false positives. Of particular interest in this thesis is another strategy referred to as weak supervision or semi-supervised learning. This approach uses a trained model to provide labels for unlabeled data, after which the model is retrained using the larger dataset [96]. Weak supervision can also be used to increase the level of abstraction of a particular dataset. [106] describes the process of turning an object detection model into a object segmentation model using Meta's Segment Anything Model (SAM[54]) by combining their outputs using a simple heuristic. Observe that one can collect weakly supervised data through the structure of the data collection process and the assistance of a model trained to recognize salient objects in the scene, but this introduces highly correlated data (which the nature of our data collection will address, in part) and that this is also a real issue for temporal data.

2.4.7 Action Recognition

Action recognition aims to detect and classify certain kinds of movement (e.g., swimming, running, waving, jumping, etc.) within image sequences. Action recognition is similar to the image classification problem extended into both the spatial and temporal domain. Image classification and object detection algorithms have made tremendous strides in accuracy and recognition rates in recent year due to neural networks. Current state of the art examples of action recognition networks include [30][114][129][35][122][38][138][28]. Figure 2.11 illustrates a number of simplified architectures common to these networks. Advances in action recognition have progressed at a much slower pace than image classification and object recognition. There are many reasons why action recognition is a much more difficult problem than the other two tasks. While there are many other factors that contribute to the complexity of the action recognition problem these two stand out among the rest.

- **Model Complexity** - The Neural Networks used to recognize actions in a video sequence are far more complex than those designed for static image classification. This complexity drastically increases the time needed to train a model, thus hindering effort to search for new network architectures (model search). In the development of these large models CPU/GPU resource limits are often ignored in pursuit of higher recognition rates and run on massive cloud-computing platforms.

- **Temporal Context** - Action recognition requires the combination of both spatial and temporal context. While spatial context can identify objects and geometric configuration, without temporal data a number of gestures described in the previous chapter that rely on motion would be unrecognizable. The motion of the camera further complicates the matter and must be compensated for in-order to make accurate predictions.

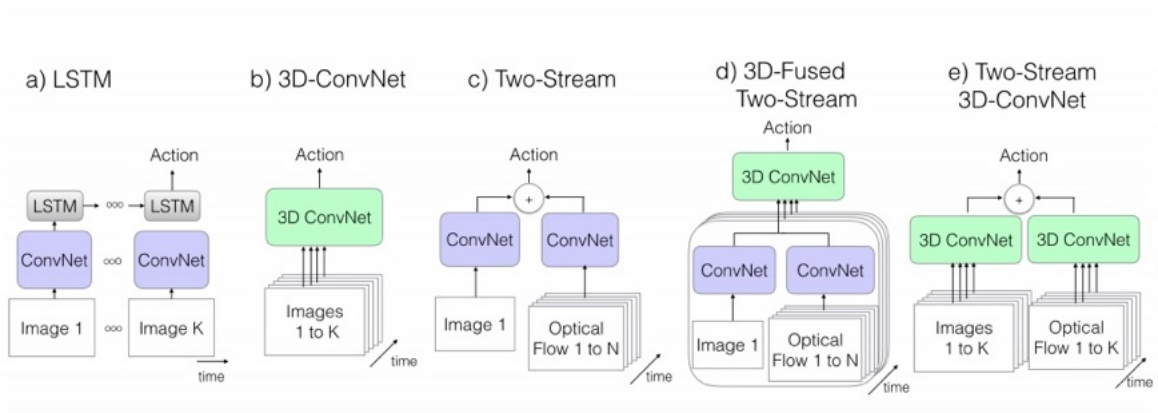


Figure 2.11: **Common Multi-stream Architectures**

Single Stream Networks

Single stream networks (such as those shown in Figure 2.11(a)-(b)) make use of a sequence of K images. In Long Short-Term Memory (LSTM [131]), action recognition models each image in the sequence is processed by an image classification CNN whose output is then feed into a LSTM model (an enhancement of the Recurrent Neural Network). After processing K images the output of the LSTM produces a probability distribution over all actions. The 3D-ConvNet[114] model processes a sequence of K images using a large CNN that integrates both spatial and temporal features into a single model.

Long short-term memory (LSTM) models are a type of recurrent nueral network (RNN) which are particularly useful for processing time series data. LSTM models are used to deal with the vanishing gradient problem[7] in RNNS, a phenomenon that can occur during training using back propagation with a gradient descent algorithm due to the use of finite-precision numbers. While LSTM's are not immune to the problem they are particularly adapt at avoiding it. The basic structure of an LSTM cell (see Figure 2.12) consists of three different gates:

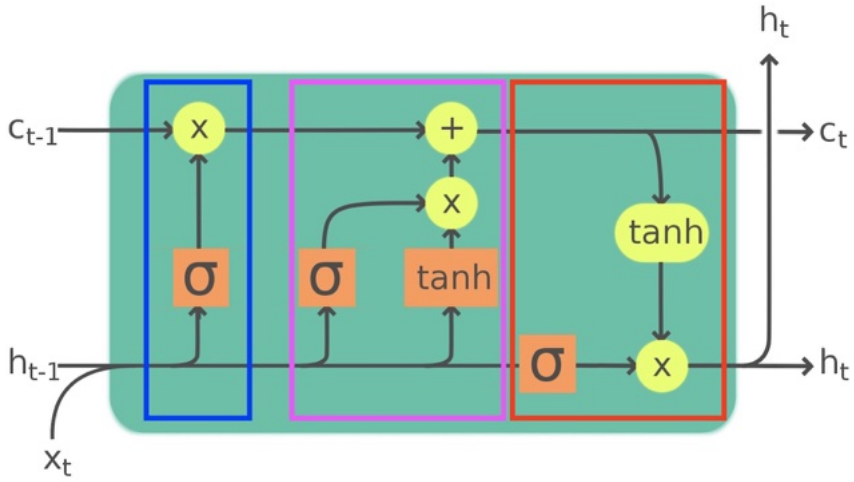


Figure 2.12: **Long Short-Term Memory (LSTM) gates**
Shows the three different gates that comprise the basic LSTM structure. Forget gate (blue, Input gate (magenta), Output gate (red).

- Forget gate - sigmoid layer determines what information to remove from the cell state using the input and previous hidden state.
- Input gate - sigmoid layer determines what information from the cell activation function to add to the cell state using the input and previous hidden state.
- Output gate - sigmoid layer determines what information from the cell state makes up the output and new hidden state using the input and previous hidden state.

Multi Stream Networks

Multi-stream networks[35][122][38][138] use a sequence of K images paired with a subsequent number of images representing the optical flow between the targeted set of images. Simple Two-Stream networks[35] use only single image and then incorporate temporal information using a sequence of optical flow encoded images.

2.5 Training in the absence of data

A perennial problem in supervised learning is the lack of a large, properly curated and labelled, set of data to train and validate the neural network. A number of different approaches have been developed to address this problem, and we review two here that are critical to this work; semi-supervised learning and the use of simulation to create a realistic dataset that can be used to train a network that performs well in the real world. This later approach is known as Sim2Real. Both approaches are reviewed briefly below.

2.5.1 Semi-supervised Learning

Semi-supervised learning (see [115]) seeks to address the issue of insufficient labelled data by combining both labelled and unlabelled training data within the training process.

The underlying concept in semi-supervised learning involves leveraging a smaller trained dataset to label a larger (unlabelled by humans) dataset, and then to train a network on this resulting set of data. Since its introduction, there have been a number of approaches to semi-supervised learning: Single classifier incremental methods continually add the most confident labels to the dataset at a set rate and then update the classifier (see [130] and [62] for example), Multi-classifier methods (see [63] and [137]) employ more that one label classifier in an ensemble fashion to score and label new data, Multi-view methods (see [12] and [121]) partition the labelled dataset to train new classifiers that are used in combination to label new data. See [115] for an example taxonomic breakdown of semi-supervised learning algorithms.

Semi-supervised learning approaches leverage the given labelled data to label a larger dataset. This can be done in one step, or in a recursive process. But central to both approaches, however, is some mechanism of clustering or assigning unlabelled data to different

labelling classes given the manually labelled data. The vast majority of previous approaches assume that the labeled dataset and the dataset being labelled share the same label set, and are in the same format. As an example of this, see [86]. Here we have a partially-labelled dataset and we seek to label other entries in the dataset with the same label set that has already been used. We observe that this might not be the most effective mechanism for problems in which one set of labels can be used to leverage an even larger set of labels through a clever organization of data collection. In terms of the clustering model of semi-supervised learning, can we simplify the clustering process, through an appropriate formulation of the data capture process, so that it can be addressed with a simpler manually labelled neural network?

2.5.2 Weakly Supervised Learning

Weak supervision [136] is an approach in which high-level sources of supervision are used to create training datasets. The primary distinction between weak and semi-supervised learning is that semi-supervised learning extends existing knowledge by labeling additional data based on already labeled examples, while weak supervision introduces new knowledge by labeling more data based on external information. Due to their similarities many semi-supervision algorithms can be easily adapted to weak-supervision by injecting this external information.

The set of potential sources for weak supervision is almost endless. Simple heuristics, complex classifiers, and even other trained networks can be used. It is also possible to integrate the output of multiple weak classifiers and use this to label the training data [126]. Perhaps the simplest form of weakly supervised learning is to use some traditional classifier (e.g., a linear classifier) to label data which is then used for training.

2.5.3 Sim2Real

Given the lack of labelled real world data to train a network, another popular approach (see [44] and [99] for examples) is to use some mechanism to simulate the data and as the labelling is known — it drove the simulation — the process of labelling the data is trivial. The fundamental problem with Sim2Real-based learning is the gap between the simulation and the real world. If this gap is too large the resulting trained network may work very well on simulated data but perform poorly on real data.

There have been many efforts to address this Sim2Real gap. For example, one can utilize simulators with high fidelity to attempt to reduce the gap, although this can be a very difficult and expensive process. See [50] and [4] for examples of the use of high performance physical and graphical simulations for Sim2Real training. See [71] for a thorough overview of sim2real techniques and taxonomic breakdown of the field.

Data augmentation can be thought of as a 'cheap and cheerful' approach to Sim2Real. Here mechanisms augment a previously labelled dataset element (e.g., rotating an image) for which the effect of the augmentation is known on the labelling. This creates simulated data with known labels. One problem with data augmentation approaches is that often the set of augmentation can be quite small (e.g., rotation or mirroring) and that properly estimating the range of parameters that can be applied to an augmentation function can be difficult to predict. For example, how much distortion can/should be applied to an image such that circle would still be recognized by a circle? For augmentation functions with limited parameterization, this may be straightforward but it will certainly not be straightforward for more complex augmentation functions.

For the problem of diver gesture sequence understanding, how can we leverage Sim2Real to develop a training dataset? One approach might be to generate fully synthetic simulated

gesture sequences. This approach would require a good simulator and a system that properly simulated the complex reality of rendering underwater scenes (see [92] and [134]). Do there exist less computationally expensive and more effective in terms of the Sim2Real gap.

2.6 Summary

The large number of efforts to develop gesture-based HRI systems speaks to the recognized desirability to be able to use gestures, either alone or as part of a multi-modal interaction strategy, to communicate with machines. Systems have been developed for almost every possible type of robot imaginable. Designers of such systems must deal with the need to actually capture the gesture information and then to understand the information encoded within it.

From a measurement point of view, systems can either be connected to the participant or observe them from a distance. For communication in the wild, the latter is clearly a more desirable approach as the application space limits the set of possible worn or connected devices that might be used. External systems can utilize either active (e.g., lidar) or passive (e.g., vision) sensing, but at some point the core problem becomes one of actually recognizing individual gestures and their sequencing.

Gesture-based languages can either be novel, application-specific or based on existing languages relevant to the task at hand. The latter leverages existing language experts and is likely to be more easily adopted for tasks as the learning barrier for humans will be lower.

Action recognition neural networks are a promising route to general purpose sign language recognition. However, to date there exist too many shortcomings that must be overcome before such architectures achieve widespread usage in real world scenarios. The massive amount of data required to train these networks for even a niche set of actions such as those

in Sports-1M[51] reveal that general action recognition requires a substantial amount of data. With the growing number of complex neural networks being used in a wide variety applications, there arises a great need for labelled data. This need for vast amounts of labelled data comes with a significant time and cost investment. Large human-labelled datasets can only realistically be produced and curated by large tech companies and large well funded research labs. Do there exist technical approaches that can be used to minimize the need for large labelled spatio-temporal datasets? Weak supervision offers a possible way around this problem as it enables a vast amount of data to be labelled in a very short amount of time. Similarly, Sim2Real provides an approach train using easily obtained data, although the sim2real gap is a well known problem. While there are certainly some drawbacks to those solutions are there ways to reduce their negative impact on the final result? This work seeks to address this question in the context of diver gesture recognition.

Chapter 3

Milton: Experimental ROV

This works requires an underwater robot with a forward facing camera and display. The Milton robot was designed and constructed to provide such infrastructure. [1]

3.1 Introduction

The field of underwater robotics is relatively small in comparison with other sub-fields of autonomous systems. To see this, contrast the number of available underwater autonomous platforms with those available for ground contact and flying robots. For both ground contact and flying robot researchers there exist a large number of experimental and application-specific platforms to support research in the lab. Unfortunately the same cannot be said for underwater robotic systems where very few such platforms exist. Interestingly, the lack of entry-level/research ROV's is not due to the lack of applications for such devices. Tele-operated underwater robots or ROVs are used extensively in the exploration of the deep

[1] An earlier version of this work was published as "Milton: An open hardware underwater autonomous vehicle"[22] in the proceedings of the 2017 IEEE International Conference on Information and Automation (ICIA), Macau SAR, China.

sea. ROV's find application in location-specific tasks (e.g., servicing underwater structures in the oil and gas industry) as well as in-long duration and long-distance operations including inspection of underwater cables and other structures.

Entry into this research domain is limited by the lack of inexpensive, extensible infrastructure to support research efforts. Here we discus the design and development of an open hardware robotics research platform nicknamed Milton[2]. Milton is designed to be extensible and to provide guidance on a number of design decisions for researchers who are interested in developing their own underwater research platform.

From a design point of view, Milton is intended to be power, data, and computation independent (it requires no tether to a surface-based operator). It is also designed to have an operational period of approximately and hour, and a maximum depth that exceeds 40m. As such, it is designed to operate within the same envelope as recreational divers. A key issue for underwater robotics research where vehicle failure may require recovery by a human operator.

3.2 Existing Systems

Recent open hardware designs developed by companies such as OpenROV and BlueRobotics have helped to introduce underwater robotics to the hobbyist and research markets. The design of these robots are adequate for simple exploration and surveying tasks, however they lack either the maneuverability, computational power, sensors or autonomous capabilities necessary for use as a research platform. From a research point of view a critical issue for many existing platforms is the lack of sufficient on-board computational power to run standard robotics middleware (e.g., ROS) and support for generic (research) sensors and

[2]Milton is named after the toast making photocopier robot from the TV show Archer.

actuators. A key design goal of Milton was the development of a platform that provides access to standard software tools (ROS) and easily extensible support for a range of sensors and computational units. Although this leads to a more bulky vehicle than might be possible the generality of the design is intentional.

There does exist a small number of commercially available ROV systems in the under $5K space that support underwater robotics research. For example, the OpenROV[82] is an inexpensive entry level vehicle capable of simple maneuvering whilst recording high definition video to a remote computer. Unfortunately due to the 4DOF motion capabilities of the robot and the lack of a standard robotic middleware it is not a viable candidate for robotics research. BlueRobotics' BlueROV[10] and BlueROV2[11] utilize high efficiency thrusters specifically designed to operate in both fresh and saltwater as well as enough thrusters to provide 6DOF motion capabilities. Unfortunately, these two designs lack the computational power and software to perform effectively without their tethers.

3.3 Milton

3.3.1 Hardware

It probably goes without saying, but a key issue in the development of any underwater robot is the requirement to keep the water out. This is an issue that is not typically the key requirement in ground-based robotics, but this requirement leads to a number of critical design decisions in the development of an underwater platform. There are many ways of meeting this requirement for an underwater vehicle, but perhaps the easiest – and the one followed in the design of Milton – is to divide the vehicle into two major sections. The first is a pressure vessel that contains components that need to be kept dry. The second is the

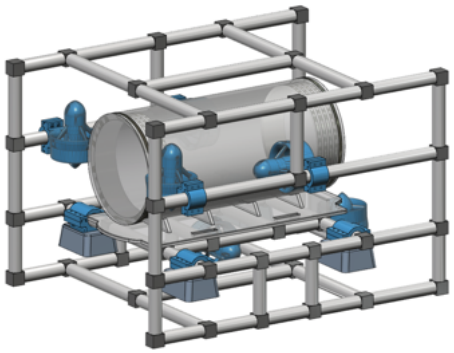


Figure 3.1: **Milton CAD Model**
Detailed CAD model of Milton showing the design of its frame and position of its thrusters, ballast weights and waterproof housing.

collection of components that can be exposed to the water. These two sections are connected by a structural lattice. For Milton, the pressure vessel is based on a cylindrical pipe structure with ends that are designed to provide a water tight seal to 40m.

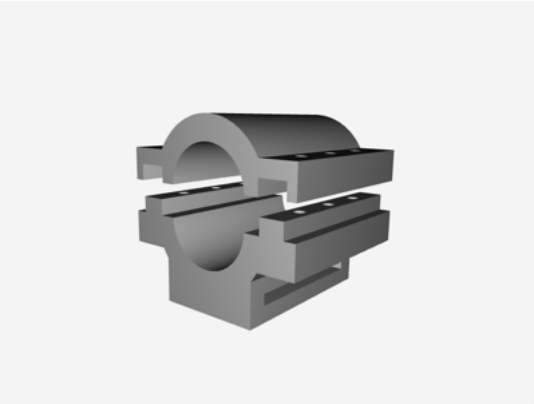
Milton makes extensive use of 3D printed parts as well as PVC pipes and fittings in order to keep production costs low and to allow for experimentation with the vehicle. Customs parts that would otherwise need to be produced in a machine shop were designed for fabrication using a 3D printer (e.g MakerBot Replicator). PVC piping and connectors are used to create the frame to which all of the robotic components are mounted. One important issues when printing 3D parts for underwater robots is ensuring that the parts themselves contain no sealed voids that might break under pressure.

The basic structure of Milton is shown in Figure 3.4. The pressure vessel is housed in the centre of the device and attachment point on the robot's frame provide connectors for thrusters, external sensors and critically ballast that is used to (i) make the robot neutrally buoyant and (ii) trim the vehicle so that it remains upright and level in the water column in

(a) battery holder tray rib

(b) standard tray rib

(c) motor bracket



(d) weight bracket

Figure 3.2: **3D Printed Parts**
A series of tray ribs provide two mounting surfaces that run the length of the housing tube. *(a) battery tray ribs provide space in the undercarriage to hold five batteries. (b) standard tray rib provides the most possible space in the undercarriage. The mounting brackets were specifically designed to firmly grip 3/4 inch PVC pipping. (c) each motor bracket has four holes with relief to attach thruster. (d) each weight bracket holds a standard dive weight and has a channel wide enough for 4cm velcro straps. This is the standard width of a recreational dive belt, making it easier to source weights to properly ballast the vehicle.*

the absence of thruster forces.

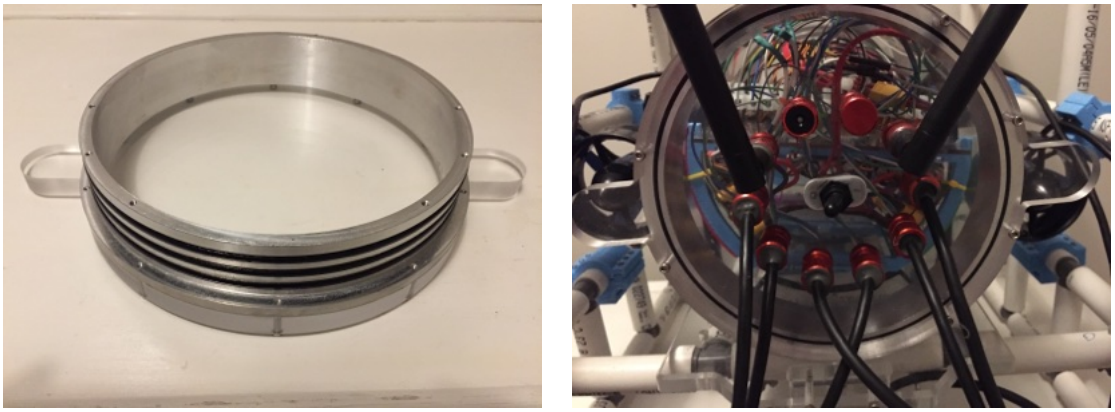
The robot's frame and it's subsequent components were designed around the requirement that it can be disassembled, packed and shipped within standard 1560 pelican cases, which can be transported by most airlines. The robot's frame consists of two side walls and three adjacent supporting structures. PVC pipes and standard wedge fittings were used to create

the frame. There is a noticeable lack of four-way pipe fittings in the design. Although it would certainly be possible to construct a vehicle with four-way connectors, such connectors are typically more fragile than their two- and three-way versions.

All sensitive electronics are compartmentalized in a single watertight housing, consisting of a 16 inch long cast acrylic cylinder (8.25 inch diameter, 0.25 thick), whose ends are secured using custom end plates. The clear end plates provide ports for visual sensors and displays. Each face plate is attached to a flange using a series of screws along the its circumference, water ingress through this gap is prevented by a single O-ring within the circumference of the mounting screws. Each flange provides a groove for three redundant O-rings which prevent water ingress into the acrylic tube. The rear face plate has a number of drilled holes to allow external non-sensitive electronic components to receive power and data. Figure 3.3(a) shows a closeup of an end-cap of the pressure vessel. A single O-ring in the upper surface provides a seal between the end cap and the end-cap assembly. These two components are mated together by a series of screws that are on the wet side of the O-ring seal. As these end-caps are inserted/removed each time the robot is serviced, redundancy here is key. The end cap plastic has extra large extensions that assist when removing the end cap from the robot. Note that there is no latch holding the end cap to the robot. Rather the robot relies on the pressure differential between the pressure vessel and the external environment to keep the end caps on the robot.

Figure 3.3(b) shows a detail of the end cap of the robot and the cables that penetrate it. Each and every penetration of the pressure vessel is a potential entry point for water. Milton utilizes Blue Robotics Cable penetrators which provide a connector with an integrated o-ring to prevent water entry along the end-cap plate and utilizes solid cables within a plastic core to eliminate water entry along the cable itself. Two end-cap penetrators require specific

attention. The first is a depth/temperature sensor (Blue Robotics Sensor MS5837-30BA) which enables the robot to estimate its depth in the water column. The second is a pressure release connector. This connector is opened when the end caps are inserted in order to facilitate sealing the vehicle. (If no pressure release was available, seating the end caps would be problematic given the sealed nature of the tube.) This opening is also used to test the seal of the device prior to deployment. By attaching a vacuum pump to the device it is possible to simulate the pressure differential that the vehicle will experience at depth, prior to deployment and to check for leaks in the vehicle's seals.



(a) blank end cap *(b) penetration end cap*

Figure 3.3: **Housing End Caps**
(a) blank end cap showing the O-ring structure. (b) end cap with cable, sensor and switch penetrators.

Electronic components within the waterproof housing are affixed to an electronics tray. This tray is constructed using a series of ribs which support two flat acrylic plates. There are two types of ribs (shown in Figure 3.2) the battery ribs are specifically designed to support five batteries firmly in place in the bottom section of the tray. The top tray is wide enough to hold the larger electronic components such as the Jetson TK1 and the ZED camera. The tray as a whole snuggly contours the interior of the cylindrical housing and is prevented from

sliding around using foam pads along the edge of the ribs.

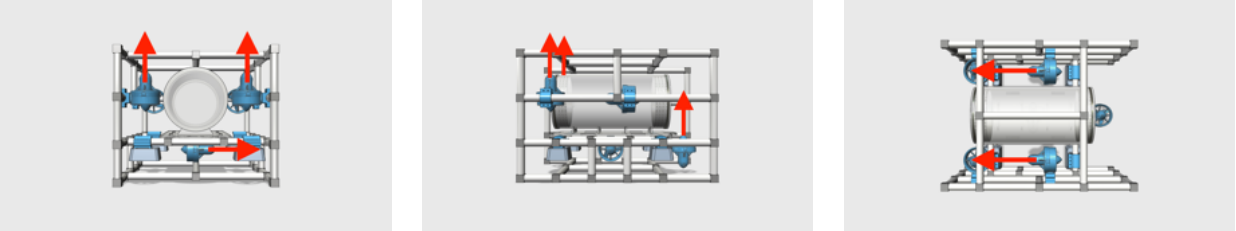
There are a number of components such as the thrusters and ballast weight that need to be rigidly attached to the robot. Instead of creating fixed mounting points for these components specialized brackets (shown in Figure 3.2) were designed that contour the PVC pipes that comprise the robot's frame. These brackets allow the thrusters and ballast weights to be mounted in a variety of places on the robot's frame.

Figure 3.4: **Milton**
shows the Milton robot fully assembled on the bench. The mounting points for the ballast weights and the 3d printed mounting points for the thrusters are shown.

3.3.2 Electronics

Electronics internal to the pressure housing has two primary functions. (i) to provide controlled power to the thrusters to move the vehicle, and (ii) conditioned power to the on-board sensors and computational units.

- **Jetson TK1** - high performance low power embedded computer.
- **3D Labs** - ZED high definition USB 3.0 stereo camera.



(a) front view *(b) side view* *(c) top view*

Figure 3.5: **Thruster Configuration**
Shows the direction of positive thrust(red arrow) for each of the motor's on the robot. Each robot is also capable of producing thrust in the opposite direction. (a) shows the motors used to roll or sway the robot. (b) shows the motors used to pitch or heave the robot. (c) shows the motors used to yaw or surge the robot.

- **Samsung 1TB SSD** - V-NAND 850 EVO solid state drive.
- **Adafruit BNO055** - 9-DOF absolution orientation sensor fused IMU.
- **BlueRobotics Bar30** - High resolution 300m depth/pressure sensor.
- **Adafruit DS3231** - High precision real time clock.
- **Adafruit INA219** - High side I2C DC current and voltage sensor.
- **BlueRobotics BasicESC** - 30A PWM electronic speed controller.
- **Teensy 3.2** - arduino-compatible micro-controller.
- **RoboSavvy Powerboard** - hot swappable battery power board.
- **MultiStaar** - 14.8V 5400mAh 10C Lithium Polymer batteries.
- **BlueRobotics T200** - high efficiency underwater thrusters.
- **Wifi Antennas** - dual-band wireless antennas.

Sensing and computation

The Jetson TK1[79] is a small embedded computer designed by NVIDIA. The TK1 has an NVIDIA Kepler GPU with 192 CUDA cores (upto 326 GFLOPS) and a 2.32 GHz quad-core ARM Cortex-A15 CPU with a typical power consumption between 1-5 watts. This is an ideal choice for an underwater robot because of its low power consumption vs high computational power. The TK1 also has an number of I/O expansion ports for connecting external sensors. There is one major flaw in the design of the Jetson TK1, when power is physically disconnected from the board a large delay is required before reconnection if the board is to initiate a boot sequence. This is problematic if a hard reset is required at depth. This was solved by removing the 0.014F capacitor at C6D4[78].

The Jetson TK1 does not provide an onboard clock to keep time while not powered on. This presents a real problem because data collected by the robot still needs to be accurately timestamped. The DS3231 real time clock provides an extremely accurate clock that is maintained while the Jetson is powered off. This ensures that data sets separated by a battery recharging step are marked as such.

The Adafruit INA219 voltage and current sensor is included as a safety sensor to prevent the robot from unknowingly running out of power in the middle of a task. This allows the robot to safely halt the current mission and safely return to the surface in the event the voltage of the main battery drops to a critical level.

The 16GB of storage onboard the Jetson TK1 is enough to store the Ubuntu operating system and software packages needed to operate the robot. However this is woefully insufficient to store data produced by the robots sensors. A 1TB solid state drive was added for data storage.

The wireless antennas enable the robot to communicate via Wifi and Bluetooth while

hovering at the surface. This allows the robot to receive commands pertaining to the current mission or to send relevant data collected from the most recent mission.

The robot's sensor suite is comprised of the 3D Labs ZED stereo camera, the Adafruit BNO055 sensor fused IMU and the BlueRobotics Bar30 depth and temperature sensor. These sensors provide the robot with both visual and inertial sensing capabilities. The ZED camera leverages the GPU on board the Jetson to produce depth maps and point clouds at 15 frames per second in VGA resolution.

Thruster control

The Teensy 3.2[89] micro-controller is used to drive the PWM signals for all six of the BlueRobotics basic electronic speed controllers. The Jetson TK1 does have PWM capabilities on four GPIO pins, however this is insufficient to control the required six thrusters. The Teensy 3.2 micro-controller was chosen due to its small form factor and compatibility with the ARM version of the FTDI_SIO kernel module. The arduino nano has a similar form factor but suffers from a defect requiring it to be physically disconnected and subsequently reconnected in order to be recognized by udev. This is problematic as any reboot would require opening the underwater housing. The Teensy 3.2 also has the benefit of not overlapping PWM pins with I2C or SPI pins allowing further expansion of its utility.

Milton uses six BlueRobotics T200 motors for thrust. These, motors are efficient underwater brushless DC motors capable of producing 15.1 Nm (Newton meters) of torque. The thrusters are configured to provided 6 degrees of freedom (DOF). The configuration and thrust pattern are detailed in Figure 3.5. Each degree of freedom is controlled by a combination of 2-3 motors surrounding the robot's center of gravity.

The two RobotSavvy power boards are used to provide power to the electronic speed

controller used to drive the BlueRobotics T200 motors. The power boards allow the seamless transition from one battery power source to a second backup battery. The power boards also include output pins to detect which battery is currently being to drive the motors.

3.3.3 Batteries

A key problem in driving any autonomous system relates to the choice of batteries. A common issue in deploying an ROV is actually transporting it to its deployment. Unfortunately international transportation places strict limits on the nature of batteries that can be transported by air, and it turns out that many island countries now lack non-air transportation. Although more bulky and powerful batteries exist, Milton relies on three Multistar 14.8V 5.2Ah Lithium Polymer batteries. Individually these batteries meet international shipping requirements, and can be easily sourced and abandoned should this be necessary[3]. The Jetson TK1 is powered from a single batteries routed through a current sensor and a buck/boost converter to condition the power to 12V regulated. Each of the other two batteries are used to provide power to three of the six thruster via the RoboSavvy power boards. These power boards provide the option of including two additional batteries as backup that can be source when the primary battery drops below a configured voltage.

3.3.4 Software

The Robot Operating System (ROS [95]) is a robotic middleware that provides a standard software infrastructure as well as a collection of tools and libraries aimed at simplifying the software development process. ROS has become the de-facto standard for research in the robotics community. Thus it is imperative that any new robotic platform geared towards

[3]batteries meet international commercial air shipping requirements at the time of writing.

research provided at least basic support for ROS.

On an embedded computer such as the Jetson TK1 computational resources are a premium commodity. Using a naive software architecture it is especially easy to consume these resource when dealing with high bandwidth sensors such as the ZED stereo camera. To help manage these resources two software stacks are defined; a control stack and a vision stack. The control stack follows the software architecture defined detailed in the ros_control[70] package. Software on the control stack is written in the form of controller plugins. These plugins can be dynamically loaded and unloaded onto the control stack using a Controller Manager. Controller plugins interact with the robot's hardware via a set of hardware resource interfaces which define the capabilities of each of the robot's sensors and actuators. These interfaces are serviced within a generic control loop feedback mechanism within the robot hardware interface layer. Hardware resource interfaces provided by control include the Adafruit BNO055, Adafruit INA219, BlueRobotics Bar30 and the six BlueRobotics T200.

The vision stack utilizes the ROS nodelet architecture[37] which allows a number of tasks to operate within a single process, without incurring copy costs when passing intraprocess messages. This architecture allows the vision stack to process a large amount of data in the form of images and point clouds without saturating network bandwidth. Much like the control stack, nodelets in the vision stack operate as plugins and can be dynamically loaded and unloaded on to the stack via a Nodelet Manager.

Chapter 4

Salient Body-part Detection

4.1 Introduction

As described in Chapter 2, a fundamental problem in utilizing a data-driven approach to recognizing a gesture-based language is obtaining an appropriately rich labelled dataset to train the recognizer. I address this problem in two phases. First, the recognition of gestures in individual frames. Here a weakly-supervised learning approach is adopted. Second, through the integration of weakly-supervised learning and Sim2Real training to label gesture sequences. Here an algorithm that splices together simulated gesture sequences using the labelled datasets from the first phase is used. I begin with the first issue, that of labelling individual frames with one of a collection of known diver gestures.

Chapter 5 deals with the problem of identifying specific SCUBA gestures underwater through a weakly-supervised process. Key to that process is the ability to identify divers and their salient body-parts in image frames. This is the problem that is considered here. This chapter describes the collection and labelling of an underwater diver dataset (SCUBANetV1+),

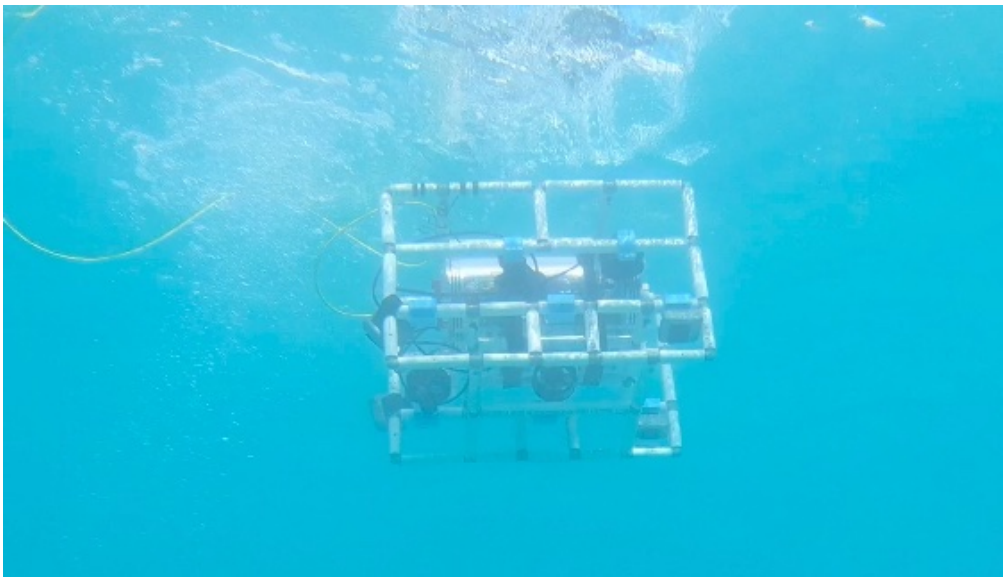


Figure 4.1: **The Milton Unmanned Underwater Vehicle (UUV)**
Milton, an unmanned underwater robotic research platform operating in the Caribbean. Milton is a thruster-based UUV which can operate using self-contained power, computation and sensing. See [22] and Chapter 3 for details on the vehicle and its design.

and the process of object detection model training and tuning using that dataset[4].

4.2 Background

Before convolutional neural networks (CNN's) became standard tools for working in computer vision, visual recognition tasks were typically based on the application of complex image descriptors that were then fed into statistical grouping processes. State of the art techniques prior to the introduction of CNN's include approaches that used image representations such as the histogram of oriented gradients (HOG) descriptor [25]. Traditional approaches such as using HOG descriptors, paired with support vector machines [84], can be effective in object detection applications. Since the introduction of Alexnet [56] in 2012 however, the subfields of

[4]An earlier version of this work was presented at the 2019 International Conference on Robotics and Automation (ICRA2019) in Montreal, Canada, as "Finding Divers with SCUBANet" [20].

Figure 4.2: **Image Samples From SCUBANetV1+**
The cold (a) and warm (b) water diver environments are very different. The colour of the water column changes from green to blue and in cold water conditions divers typically wear substantially more thermal protection: including a hood, heavy insulated gloves and a full body drysuit. In warm water conditions such protective clothing is often minimal and hoods are rarely worn. Note however that in both conditions divers wear face-masks which changes the problem of face detection from its terrestrial counterpart.

image classification and object detection within computer vision have advanced significantly through the adoption of convolutional neural networks and other data-driven approaches. A core problem in the application of supervised CNN's to a new domain is the need to design an appropriate network architecture and its training. A number of software tools (e.g., Tensorflow [1], Caffe [48]) have been developed to facilitate the creation of CNN models. In addition, the publication of pre-trained CNN's has enabled a concept known as "the transfer of learning" from psychology, to be adapted to deep learning. In the literature (see [67]) this technique is commonly referred to as "transfer learning" or "retraining". Transfer learning enables the reuse of a state-of-the-art model trained on a large-scale dataset with numerous classes to be quickly retrained to recognize a different set of classes. This process typically replaces the final classification layer of a pre-trained CNN with a new one, and in retraining only the new output layer is updated and the rest of the model remains unchanged. As a result transfer learning drastically reduces the barrier to entry in the field of object detection for any specific task.

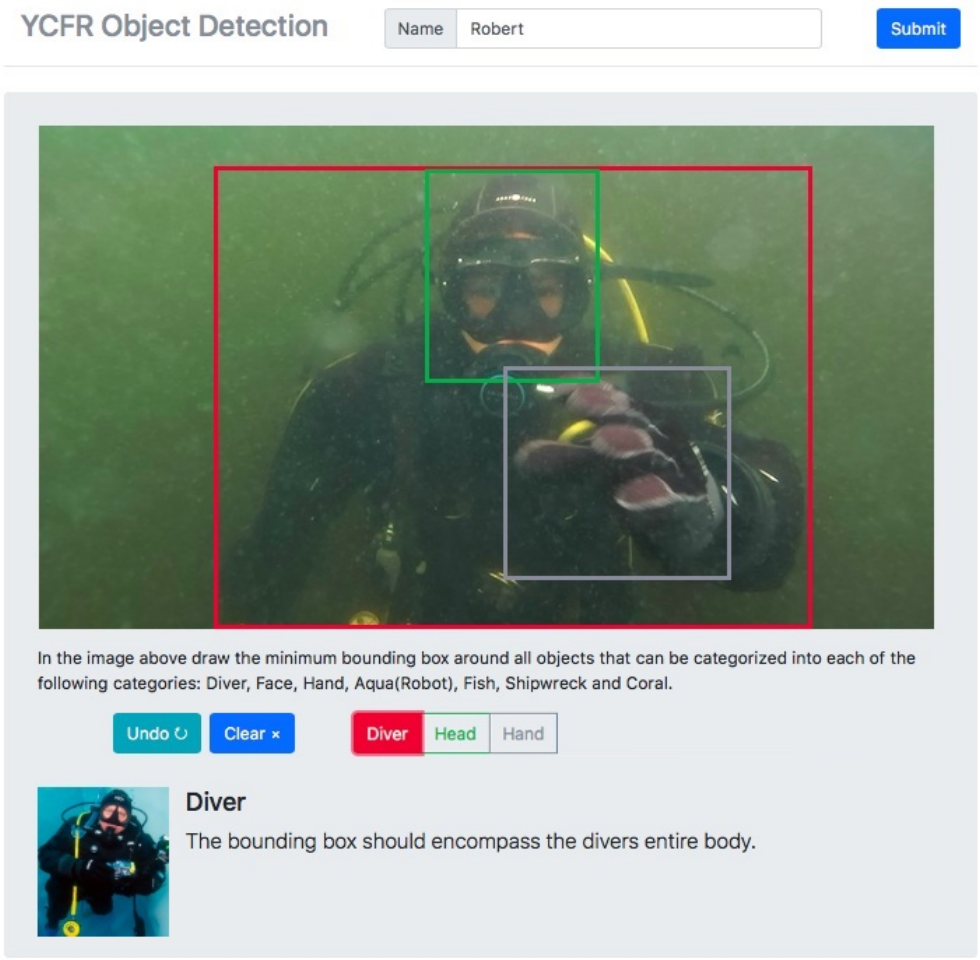


Figure 4.3: **Webturk**
The web-based platform for crowd sourcing ground truth data annotations. The user selects and annotates specific features using a browser-based tool.

Deploying a CNN to the problem of diver-robot communication through standard diver hand signs using transfer learning requires an appropriately trained dataset and an appropriately pre-trained CNN that can be adapted to the problem at hand. Given the nature of the standard diver gesture-based communication strategy a first step in this recognition process is the detection and localization of key aspects of this gesture – the diver, their head and hands and we begin with the problem of recognizing these features in this domain.

4.3 Data Collection

Images within the SCUBANetV1+ dataset were collected using a custom unmanned underwater vehicle (UUV) named Milton [22] (see Figure 4.1 and Chapter 3) as well as through the use of standard underwater cameras. This robot was developed as a research platform to investigate the potential for gesture-based human-robot interaction in an underwater environment. Milton is a thruster-based underwater robotic with 6-DOF motion capabilities, high definition stereo vision, an inertial measurement unit and an underwater depth and temperature sensor. Milton is self-contained for power and computation, has an operating depth of over 60m and can operate autonomously for over an hour.

The SCUBANetV1+ dataset contains images from both freshwater and saltwater environments. The freshwater portion of the dataset was collected in Lake Seneca in King City, Ontario, Canada. The saltwater portion of the dataset was collected just off the west coast of Barbados north of Holetown. Sample images from the dataset are shown in Figure 4.2.

The SCUBANetV1+ dataset contains both left and right images from Milton's onboard stereo camera. While left and right images are not used in tandem to discern depth information from the scene in the work presented here, different perspectives of the same scene can help to increase the robustness of the training process. To further increase the diversity of images in the dataset, divers were asked to present different hand gestures to the robot while it was recording, this was done to help prevent a single hand pose from being over represented in the set of collected data.

	Diver Labels	Head Labels	Hand Labels	Total Images
Warm Water	1109	951	1307	1387
Cold Water	572	597	649	681
Total	1681	1548	1956	2068

Table 4.1: **Data Statistics**
Distribution of image annotations are reported for each environment across all classes. Total annotations across all classes are reported along with a final tally for all images in the dataset.

4.4 Data Annotation

The set of images obtained from the data collection stage produced an extraordinary amount of data as the two cameras have a native resolution of 1280x720 and operate at 15hz. With such a large collection of images it is unreasonable for one person to annotate the entire dataset. Such a strategy would also open up the potential for labelling bias given the small number of individuals labelling the data. A common solution to this problem is crowd sourcing the annotation of data. MechanicalTurk[116] is a popular crowd sourcing platform for outsourcing ground truth annotations, and this was the process followed here.

The entire unlabelled dataset was stored on a host machine running a simple web-server. Clients (aka turkers) were provided with a URL that presents them with a single image at a time that is annotated using the simple browser-based interface shown in Fig. 4.3. Once annotated and submitted a new random image from the dataset is presented to the turker. Cookies track the user's list of annotated images, preventing redundant work. This does not prevent the possibility that multiple clients may provide annotations for a single image but such accidental duplication is rare given the size of the dataset. Deliberate duplication to aid in data cleaning can be helpful, as described below.

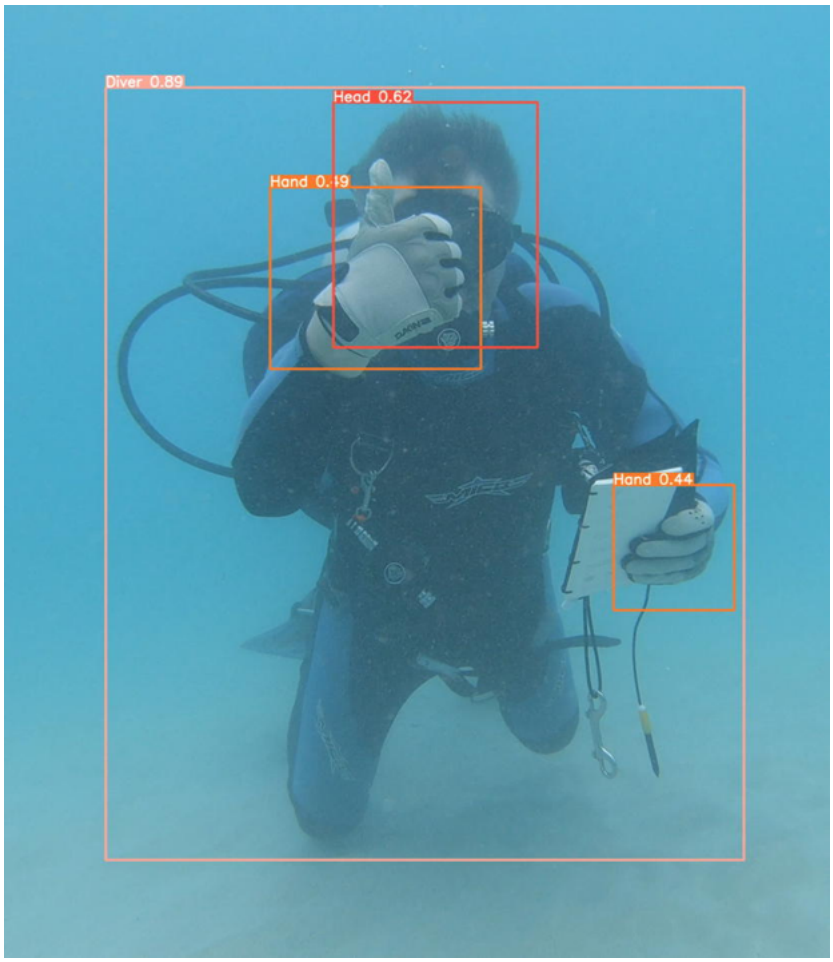


Figure 4.4: **SCUBANetV1+ Detections**
Results of running the SCUBANetV1+ detector to identify hands, bodies, and heads.

4.4.1 Data Cleaning

The data annotation process provides a simple mechanism to cull images from the final dataset. Clients have to option to mark images as "Empty/Corrupt" or submit an image without annotations. These annotations mark an image as unsuited for inclusion in the final curated dataset. Sanity-checking procedures were applied to the annotations returned include removing bounding boxes that are either too small or out-of-bounds.

Annotations for this dataset were crowdsourced, and the user base was open and unrestric-

tive. Recognition abilities may differ user to user, some users may make mistakes, some users might not fully understand the task, and in rare cases are purposefully nefarious. The best way to prevent these problems from appearing in the ground truth data is to have multiple annotations from different sources that can be coalesced into a single annotation. A number of methods for identifying bad labels or data decontamination are described in the literature. See [32] and [5] for examples. These approaches suffer from not being able to distinguish between harmful mislabeled data and challenging properly labeled data. The solution to this problem is simple, keep and train on multiple ground truths per image (if available). This approach relies on three observations. The first being that easily recognized objects are often labeled with minor bounding box variations. The second is that hard to recognize objects are labeled less frequently with moderate bounding box variation. The third is that nefarious turkers are rare and can be screened for by using properly seeded validation samples. Keeping these observations in mind, training on multiple (possibly conflicting) ground truths per image can have some benefits. The impact of nefarious labels is degraded due to their low prevalence within identical training images. Multiple similar labels enforce the importance of recognition over bounding box placement.

Screening for harmful labels is difficult and algorithmically complicated. However nefarious users can be screened for by seeding the same N images to each user in their first K (N less than K) annotation steps so as to asses the users abilities. The users annotations for these N images are compared to properly curated ground truth data and if they do not score at least 90% mAP@[0.50-0.95]IOU all annotation from this user are discarded. mAP@[0.50-0.95]IOU is defined as the mean of the average precision (mAP) across all class label where the intersection over union (IOU) or overlap between the detection box and the ground truth lies within the range 0.5 to 0.95.

Given the crowd-sourced nature of the dataset, the set of annotated images and its magnitude evolves with time. The dataset consists of approximately one hour of imagery from the left and right cameras broken into subsets as described in Table 4.1. When the server was taken offline, of the approximately 120,000 images (15hz x 2 cameras x two hours) in the dataset, over 2,000 had been annotated, with the number of labelled diver, diver head, hands for each subset given in Table 4.1. Of these 2,000+ annotated images, 85% of them were selected for the training set, while the other 15% were used as an evaluation set.

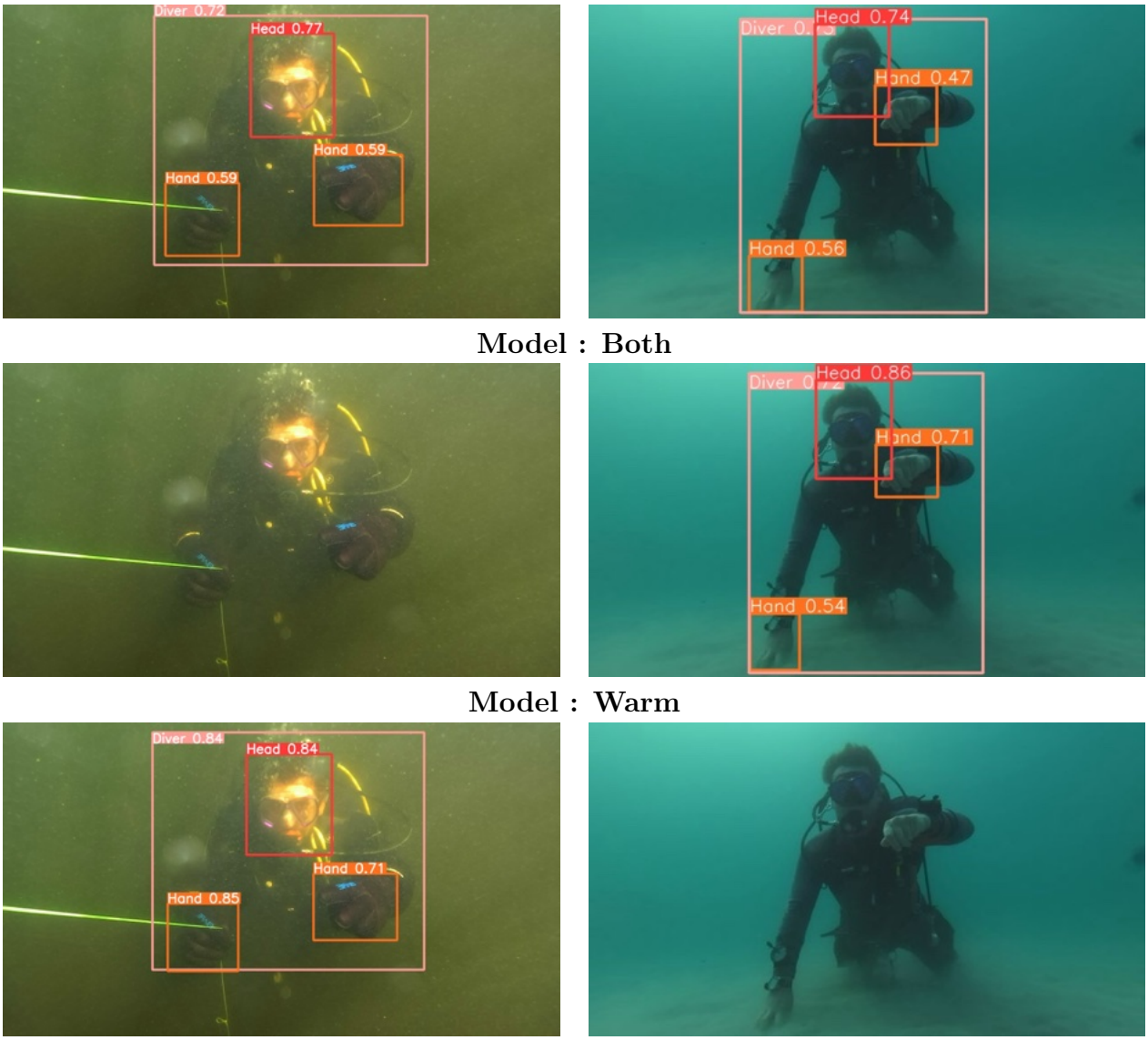
4.5 Model Training

The earlier version of this work described in [20] utilized Reznet and InceptionV2. Here a more modern architecture is used. The Yolo architecture[98] (specifically YoloV8m) was leveraged for its cross-platform support, scale-ability, performance and ease of use. The 'm' (medium) Yolo model was chosen given the computational resources available and the size of the training dataset. The pre-trained YOLOv8m model was striped of its final classification layer and retrained using a new fully connected layer corresponding to the number of classes in SCUBANetV1+. As the SCUBANetV1+'s dataset is broken down by where the data was collected (cold water in dry suits and warm water in wet suits), three different detectors (both, warm, and cold) were trained and evaluated on test datasets drawn from the same three conditions. Results are given in Table 4.2.

Figure 4.4 shows typical labelling results using SCUBANetV1+. Of particular interest is the performance in warm water as the gesture dataset (discussed in Chapter 5) was collected under those conditions. Best results of the three different SCUBANetV1+ detectors on the warm dataset are highlighted in bold in Table 4.2. Figure 4.5 shows the output of a sample run of the cold, warm and both recognition models on an image from the cold and warm

Train	Test	Class			
		All	**Head**	**Diver**	**Hand**
	Cold	0.554	0.535	0.688	0.441
Cold	Warm	0.0142	0	0.0425	0.000035
	Both	0.211	0.229	0.306	0.0977
	Cold	0.0778	0.1135	0.0897	0.00896
Warm	Warm	**0.511**	0.507	**0.669**	0.**357**
	Both	0.362	0.367	0.484	0.234
	Cold	0.560	0.574	0.665	0.441
Both	Warm	0.501	**0.514**	0.647	0.341
	Both	0.521	0.537	0.651	0.375
Tuned	Warm	0.518	0.536	0.679	0.339

Table 4.2: **SCUBANetV1+ Performance Comparison**
SCUBANetV1+ mAP@[0.50-0.95]IOU values for baseline networks trained and evaluated on cold, warm and both datasets. Best performance in a warm test environment is in bold. The cold network does not perform particularly well on either the Both or Warm datasets. Also shown is the performance on the Warm dataset of the best Warm model after hyperparameter tuning.



(a) Domain:Cold (c) Domain:Warm

Figure 4.5: **Example Results from SCUBANetV1+ Domains (cold, warm, both)** Note that the cold model was unable to identify any labels in the warm dataset example shown. Nor was the warm model able to identify labels on the cold dataset example shown.

datasets. Of the three models, the warm model performs most consistently well over the various body parts classes of interest for the wam water dataset. This warm water model was chosen for additional tuning before using it for the weak supervision task described in

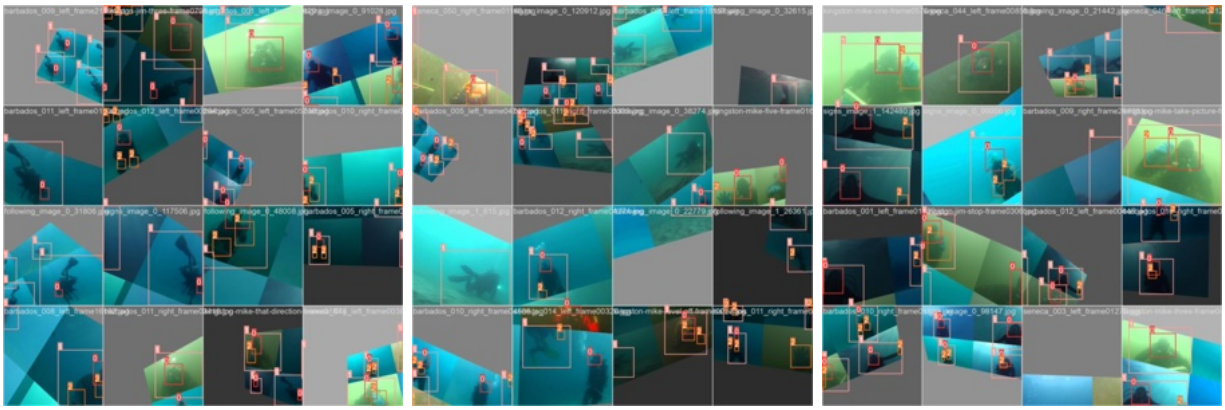


Figure 4.6: **Augmented Image Samples From SCUBANetV1+**
Images fed into the model for training are markedly different from those in the original dataset. Multiple changes in each image have been made to varying degrees.

the next chapter.

4.6 Tuning

The training of a Yolo model can be adjusted by setting different hyperparameters. How each of these parameters effect the performance of the resulting model depends on the evaluation metric that you wish to maximize as well as the objects being detected and the composition of the training and test datasets. This means there exists no global set of guidelines for manually tuning these parameters to improve the performance of a model. One simple way to perform hyperparameter tuning is to sample the parameter space and perform a number of different training runs, selecting the set of parameters that performs best. The parameter space used here (shown in Table 4.3) was choosen as it is the default parameter space that Ultralytics suggests for tuning YoloV8 models. The first eight parameters change how the training algorithm updates parameters within the model. The next thirteen parameters change how training images are augmented each time they are used, Fig. 4.6 depicts a sample set of training images that have been augmented in various ways.

Parameter	Value	Description
lr0	0.04254	initial learning rate $\in [1e-5, 1e-1]$
lrf	0.02576	final learning rate (lr0 * lrf) $\in [0.01, 1.0]$
momentum	0.7677	SGD momentum $\in [0.6, 0.98]$
weight_decay	0.0005707	optimizer weight decay $\in [0.0, 0.001]$
warmup_epochs	3.1562	warmup epochs $\in [0.0, 5.0]$
warmup_momentum	0.2890	warmup initial momentum $\in [0.0, 0.95]$
box	0.04434	box loss gain $\in [0.02, 0.2]$
cls	0.4116	class loss gain $\in [0.2, 4.0]$
hsv_h	0.02408	image HSV-Hue $\in [0.0, 0.1]$
hsv_s	0.006433	image HSV-Saturation $\in [0.0, 0.9]$
hsv_v	0.6358	image HSV-Value $\in [0.0, 0.9]$
degrees	35.35	image rotation (+/- deg) $\in [0.0, 45.0]$
translate	0.4103	image translation $\in [0.0, 0.9]$
scale	0.6034	image scale (+/- gain) $\in [0.0, 0.9]$
shear	1.481	image shear (+/- deg) $\in [0.0, 10.0]$
perspective	0.0006267	image perspective $\in [0.0, 0.001]$
flipud	0.0	image flip up-down (prob.) $\in [0.0, 1.0]$
fliplr	0.3812	image flip left-right (prob.) $\in [0.0, 1.0]$
mosaic	1.0	image mosaic (prob.) $\in [0.0, 1.0]$
mixup	0.0	image mixup (prob.) $\in [0.0, 1.0]$
copy_paste	0.9650	segment copy-paste (prob.) $\in [0.0, 1.0]$

Table 4.3: **Yolo Hyperparameters**
Four significant digits shown.

The results of hyperparameter tuning that produced optimal results on the warm dataset when selecting for mAP@[0.50-0.95]IOU are shown in the second column of Table 4.3. The F1 scores for the final trained model are shown in Fig.4.7. These value were calculated by computing the harmonic mean of the per class precision and recall scores as reported by Yolov8 and evaluated on the testing dataset. The F1 score for the hand class is relatively low with respect to the diver and head class, this is not concerning as false negatives will only reduce percentage of data that gets labels. . This is a measure of the precision of the placement of the predicted bounding box in comparison to the ground truth bounding box as a percentage on the area of their intersection divided by the area of their union. This metric

Diver	Head	Hand
0.879	0.915	0.714

Figure 4.7: **ScubaNetV1+ F1 Scores**
Shown are the F1 scores for SCUBANetV1+ on each of the hand gesture classes after hyperparameter tuning.

was chosen because in the weak supervision labelling task this model is utilized to seed a deep neural network (see Chapter 5) in the recognition of a pre-defined vocabulary of hand signals, and bounding box placement will be its primary task.

4.7 Summary

SCUBANetV1+ is a transfer-learning based detector for SCUBA diver parts, that uses YoloV8m as a base detector. Three different detectors were developed from cold, warm, and both datasets. Evaluation on warm data – of particular interest in this work – determined that the detectors trained on the warm dataset was the most accurate. After hyperparameter tuning this detector is used in SCUBANetV2 in a weakly supervised structure as described in the next chapter.

Chapter 5

Gesture Recognition

5.1 Introduction

SCUBA divers operating at depth have developed a standardized applied gesture-based communication language that can be leveraged to enable communication underwater, but it would be expensive and perhaps impractical to develop a hand-labelled dataset of these gestures to support a machine learning-based approach to the task. There exists a large set of symbols, and it would be necessary to used trained divers to do the labelling. To avoid the cost of hand labelling such a large dataset, here the process of generating a labelled dataset is semi-automated. The SCUBANETV1+ model (discussed in Chapter 4) is used to identify salient objects (divers, their heads and hands) in images, and then a weakly supervised learning process is used to label the complex set of diver gestures. The result of this process is a system that can recognize a large number of diver hand gestures. This chapter describes this weakly supervised learning approach. Performance of the resulting system is compared against a hand-labelled set of diver gestures[5].

[5]An earlier version of this work was presented at the 2023 International Conference on Robot and Human Interactive Communication (RO-MAN2023) in Busan, South Korea, as "Recognizing diver hand gestures for

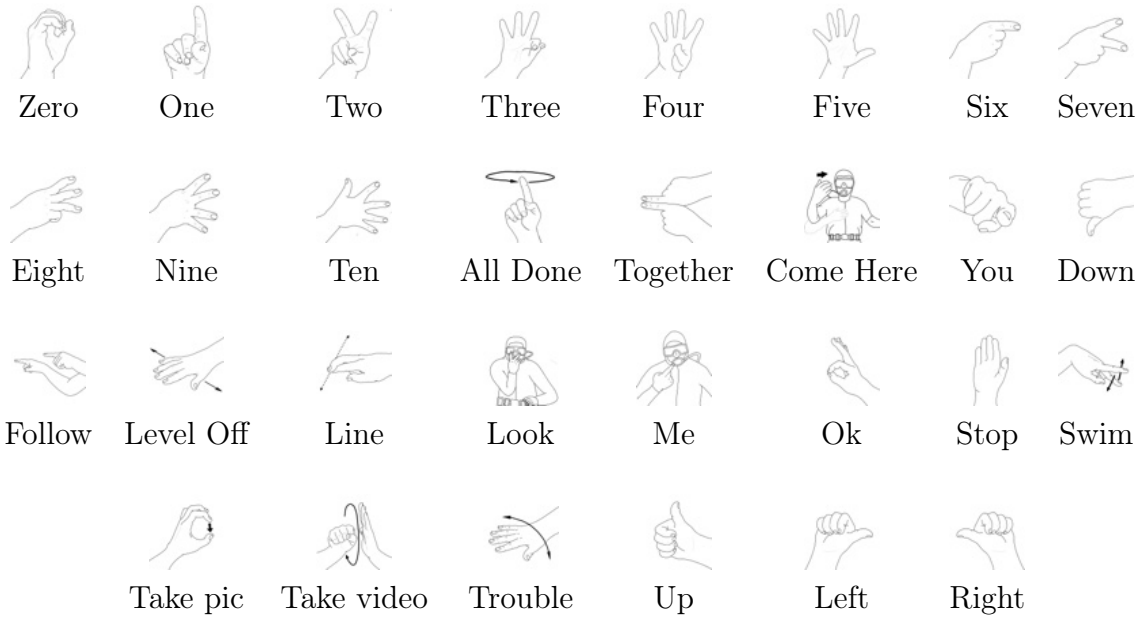


Figure 5.1: **Underwater Hand Sign Gestures**
These are drawn from the set of hand gestures normally used by recreational and commercial SCUBA divers. The gestures can be performed with or without gloves.

5.2 Background

Recognizing the limitations imposed by the underwater environment, divers have developed a strategy for communicating underwater. A standard set of hand gestures (signals) have been developed[24]. These symbols are shown in Figure 5.1. Communication involves sentences made up of these hand sign gestures. Work to date in human-robot communication has generally ignored the possibility of leveraging established diver to diver hand signal communication gestures, and instead has concentrated on the development of novel gesture languages. For example [47] describes a rudimentary set of one handed gestures pairwise combinations of these gestures map to a single command, while [19] describes a limited syntax underwater communication language loosely based on standard diver signals.

human to robot communication underwater" [21].

In order to obtain a better understanding of the scope of the problem in the context of underwater communication, Figure 5.1 provides an overview of gestures that are commonly used in underwater communication and which are likely to be useful for human to robot interaction at depth. There are many other communication symbols that are also used, but the set shown in Figure 5.1 has been limited to gestures that might be useful for human to robot communication. For example, gestures related to diver safety and gestures used to identify marine wildlife that are commonly used in recreational diving have been omitted.

Diver signals are designed intentionally to be discriminated easily, and are either static presentations or are composed of very simple animated gestures. Even so, a minimal target of 500 samples of each class would require 15,000 labelled images. Hand labelling this imense dataset would be a time consuming and arduous task, and a task that will only become more time consuming and arduous if the set of gestures was expanded to include signals for safety-related signals and signals related to specific environments or tasks. Futhermore, it would be necessary to obtain labellers who are proficient in recognizing SCUBA diver gestures. Instead of following this manual labelling approach the capabilities of SCUBANetV1+ are coupled with a structured data collection process to avoid complex hand labeling. This weakly-supervised approach integrates structured data collection with the part labelling model discussed in the previous chapter.

5.3 Data Collection

Diver gestures were collected in warm water near Holetown, Barbados. Data collection and storage was approved by the York University ethics committee for research associated with experiments involving humans (Certificate e2022-161). Participants signed an informed consent form as well as an image and video release form. Data collection involved having

Figure 5.2: **Temporal Labelling of a Gesture Sequence**
Each video sequence is segmented manually into portions that are presenting the gesture or idle/null. An idle label indicates that the diver is idle, while null indicates that the diver is engaged in an activity unrelated to data collection and this frame should be ignored.

divers operating at depths between 15'-25' supervised by licensed commercial divers. Data was collected using either the Milton robot shown in Figure 3.4, or using standard underwater video cameras. Data collection for individual gestures was collected by presenting a visual cue for a specific gesture (e.g., "Up") and then having the participant repeat this gesture several times while returning to an "Idle" state between presentations of the gesture. As each diver's presentation of the gesture varied, and as data collection was performed underwater under natural circumstances, it was not possible to obtain exactly the same number of frames of the gesture from each presentation. The pose of the camera while recording data varied as well, helping to ensure data was collected under a variety of different conditions. Six divers participated in data collection which took place over a number of different data collection sessions, under a range of different underwater and surface conditions. All data collection took place under available natural lighting.

5.3.1 Data Annotation

Each data sequence corresponds to a specific gesture and upon return to shore was segmented manually. The basic segmentation task is illustrated in Figure 5.2. The video sequence was decomposed into segments that indicate that the diver is performing the task associated with the gesture, "Up" in the case of example given in Figure 5.2, or that the diver was idle (the

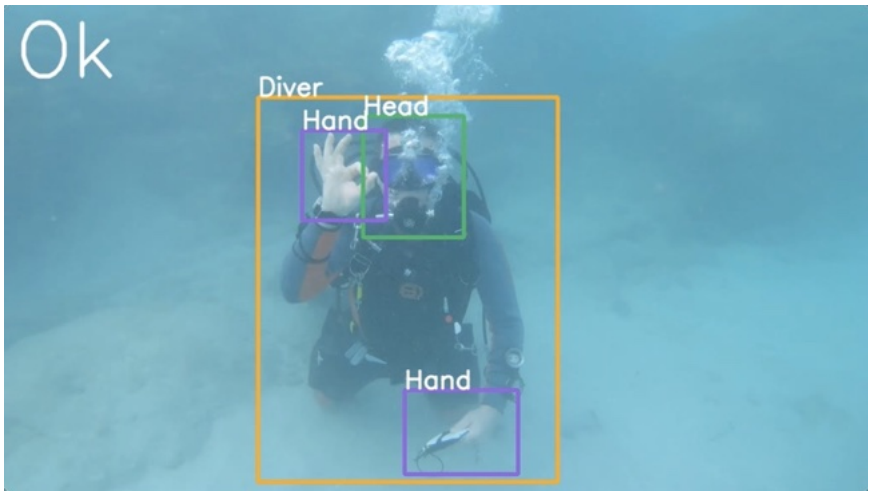


Figure 5.3: **Manual and Generated Labels**
The gesture label for this particular frame is displayed in the top left corner also displayed are the results of running the SCUBANetV1+ detector to identify hands, bodies, and heads.

divers hands were not being used for communication), or that the diver was performing some other task unrelated to data collection (these frames were excluded from the dataset). As the cameras being used to collect the gestures operate at 30 hz, a large number of image frames are captured for each gesture.

5.4 Dataset Generation

Knowledge of the nature of the gesture coupled with the structure of the data collection enables the trained SCUBANetV1+ detector to label specific gestures. The structured data collection coupled with SCUBANetV1+ enables automatic segmentation of hand labels into one of the pre-defined gesture classes. Figure 5.3 illustrates the basic process. This shows the SCUBANetV1+ identified class labels and bounding boxes for head, body and hand from a frame of a diver presenting the "Ok" gesture. Given the structure of the data collection process, the diver is known to be making the "Ok" gesture in this frame, and a labelled

Gesture Class	Heuristic
All	Identify dominant hand as the larger/higher hand Handedness of the dominate hand is determined relative to the off-hand and body center line Bounding boxes are enlarged slightly
Idle	If one hand is idle the label both hands as idle
Single-handed	Dominant hand gets the label
One-Five	Dominant hand bounding box is expanded upwards
Six-Ten	Dominant hand bounding box is expanded based on handedness
Look	Dominant hand and head bounding boxes are merged
Two-handed	Hand bounding boxes are merged and expanded based on handedness

Table 5.1: **SCUBANetV2 Heuristics.** The heuristics used to map hands to gestures.

bounding box of the "Ok" gesture can be produced for this frame without having to manually label the bounding box. In general, a human need only segment frames of a video into either displaying the target gesture or "Idle". The SCUBANetV1+ detector can than take this image classification data and combine it with *a priori* knowledge of the salient gesture in the scene to produce a more refined label.

The existence of more than a one detected hand in the image presents a labeling conflict when both hands are detected. To resolve this a simple height-based heuristic is used to characterize the dominant hand. Two handed gestures (e.g., together) are labelled with their union as a single label. Table 5.1 summarizes the heuristics that are applied depending on the gesture classification assigned to the frame. The resulting dataset is referred to as SCUBANetV2.

The SCUBANetV2 dataset consists of frames of a set of 32 labelled hand gestures (the 30 in Table 5.2 plus "Idle" and "Off Hand" labels) including their bounding boxes and the bounding box of the diver's head. Generating the SCUBANetV2 dataset proceeds as follows: Individual frames from each video clip are extracted. Frames are then processed using the SCUBANetV1+ detector to detect the head, body and hands of all divers in the image.

At this stage consistency checks are applied to ensure the image is a good candidate for inclusion in the dataset. The image must have one diver, one head and one or more hands. If multiple detected hands overlap only the largest is retained, the remaining are ordered by their confidence values and only two are kept for further labeling.

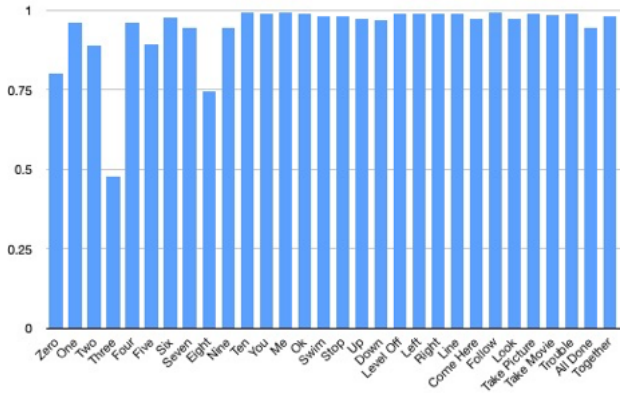
The process of running the SCUBANetV1+ detector with the gesture classes obtains a large set of labelled data for further training (see Table 5.2). The total number of labelled images exceeds 290,000. For non-idle labelled hands, the hand not being used to provide gestures is labelled as "Off-Hand". In addition to the set of hand labels, this process also obtains over 140,000 hands labelled as not belonging to one of the classes identified in Table 5.2. These hands make up the "Idle" class. There are also over 290,000 examples of labelled 'heads'. To normalize the sizes of the classes, for classes with more than 5000 elements, only 5000 elements were chosen randomly to include in the SCUBANetV2 dataset.

5.5 Model Training

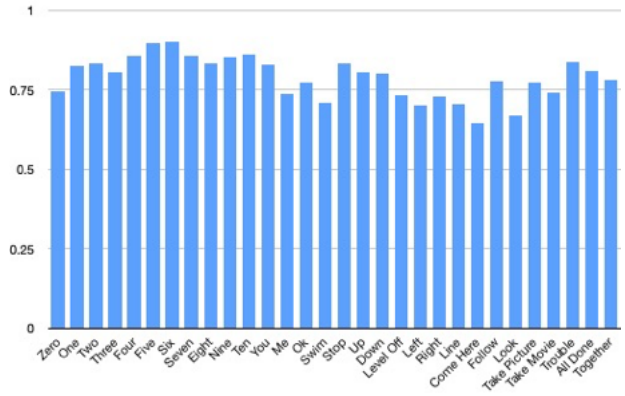
The dataset is randomly split into 70% train, 15% test and 15% validate sets. Table 5.2 summarizes the weakly supervised data collection process by gesture class. Transfer learning from YOLOv8m was used to train the SCUBANetV2 detector for the labeled classes described in Figure 5.1 plus the idle, off-hand and head classes. This resulted in a total of 33 classes. Training was for 20 epochs and initially used the default YOLOv8 hyperparameters.

5.6 Tuning

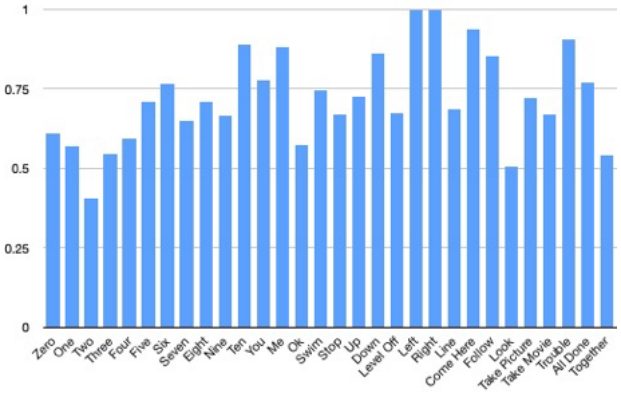
This YoloV8 model underwent the same hyperparameter tuning from initiall hyperparameters as described in Chapter 4, sampling the parameter space 100 times. The results of hyperpa-



(a) Label performance



(b) Idle performance



(c) Label expansion

Figure 5.4: **Validation of SCUBA Gesture Recognition.** Over 240,000 frames that included both frames involved in training and novel but highly correlated frames were used for validation. (a) shows fractional labelling rate for the various gesture labels for the gesture classes given in Table 5.2. Performance on labelled frames was very good. (b) shows fractional labelling rate for idle frames for a given gesture type. (c) shows fraction of idle labelling errors that recognized the idle gesture as belonging to the specific gesture that was being demonstrated in that training set. The network generalized the appearance of gestures and was identifying idle frames on the shoulder of a set of labelled frames as expressing that label.

Gesture	S	F	L	Gesture	S	F	L
Zero	8	8609	2519	Down	9	9302	3633
One	12	11160	852	Follow	7	9017	3780
Two	9	7758	2526	Level	8	9945	4386
Three	7	6528	2164	Line	7	9592	4725
Four	10	9850	3319	Look	6	4843	1637
Five	9	9413	2767	Me	8	8878	3368
Six	11	9991	3753	Ok	8	8904	3008
Seven	12	12006	3712	Stop	9	10336	3244
Eight	11	9614	4142	Swim	10	10921	4424
Nine	10	9033	3589	Take Pic	10	9914	2434
Ten	10	8999	3319	Take Movie	10	11160	6005
All Done	7	9223	4947	Trouble	8	9532	3946
Together	9	12448	3998	Up	9	9746	3823
Here	8	8158	3665	You	8	9631	2740
Left	16	17658	3313	Right	15	16836	2429

Table 5.2: **SCUBANetV2 Dataset**
For each gesture; the number of video sessions that were collected (S), the total number of frames of that gesture that were identified (F), and the total number of labelled gestures that were recovered (L). The 281 files that were processed also provided 140,000 examples of hand gestures that did not fall into these categories (known as 'idle gestures').

rameter tuning determined that produced optimal results when selecting for recall(B) are shown in Table 5.3. This is a measure of the ability of the model to successfully detect the presence of classes in the dataset. This metric was chosen because it is more important for this model detect the presence of a gesture than to accurately place the bounding box. The F1 scores after hyper-parameter tuning are shown in Fig. 5.5. These scores were calculated using the same method described in the previous chapter. Overall the scores are good, however the two-handed gestures "Take Movie" and "Follow" under perform the rest which could be improved by changes the heuristics used to combine weak labels.

Parameter	Value	Parameter	Value
lr0	0.03707	**momentum**	0.04685
lrf	0.07168	**weight_decay**	0.0002473
box	0.04107	**warmup_epochs**	3.8220267587432795
cls	0.8163	**warmup_momentum**	0.2890
hsv_h	0.01601	**degrees**	0.2320
hsv_s	0.3550	**translate**	0.5267
hsv_v	0.8080	**scale**	0.2038
shear	3.8875	**perspective**	0.0009383
flipud	0.0	**fliplr**	0.3373
mosaic	1.0	**copy_paste**	0.9689
mixup	0.0		

Table 5.3: **Yolo Hyperparameters**
Four significant digits shown.

5.7 Validation

As the goal is to recognize specific gestures, rather than the exact location of the gesture we concentrate here on the ability of the weak supervision to properly label the gestures. The process of choosing frames for training the SCUBANetV2 gesture detector involved selecting random frames from the large number of labelled video frames. The SCUBANetV2 dataset contains considerably more labelled data than the training set. Given this, the detector was evaluated on the entire SCUBANetV2 dataset less the portion that was used in training. This includes both Idle frames as well as gesture frames. The evaluation set contains over 140,000 idle frames and over 104,000 labelled frames. Results of this validation process are provided in Figure 5.4. Figure 5.4 plots percent correctly labelled frames for both frames containing gestures (Figure 5.4(a)) and frames that were labelled as being idle (Figure 5.4(b)). For frames containing the appropriate gesture, SCUBANetV2 is quite successful in identifying the gesture with recognition rates in the 80%-95% range for each gesture. At first viewing, the percent correct labelling for idle frames is not as promising, with success rates in the 50%-90% range. A review of the labels returned by SCUBANetV2 demonstrated an interesting pattern.

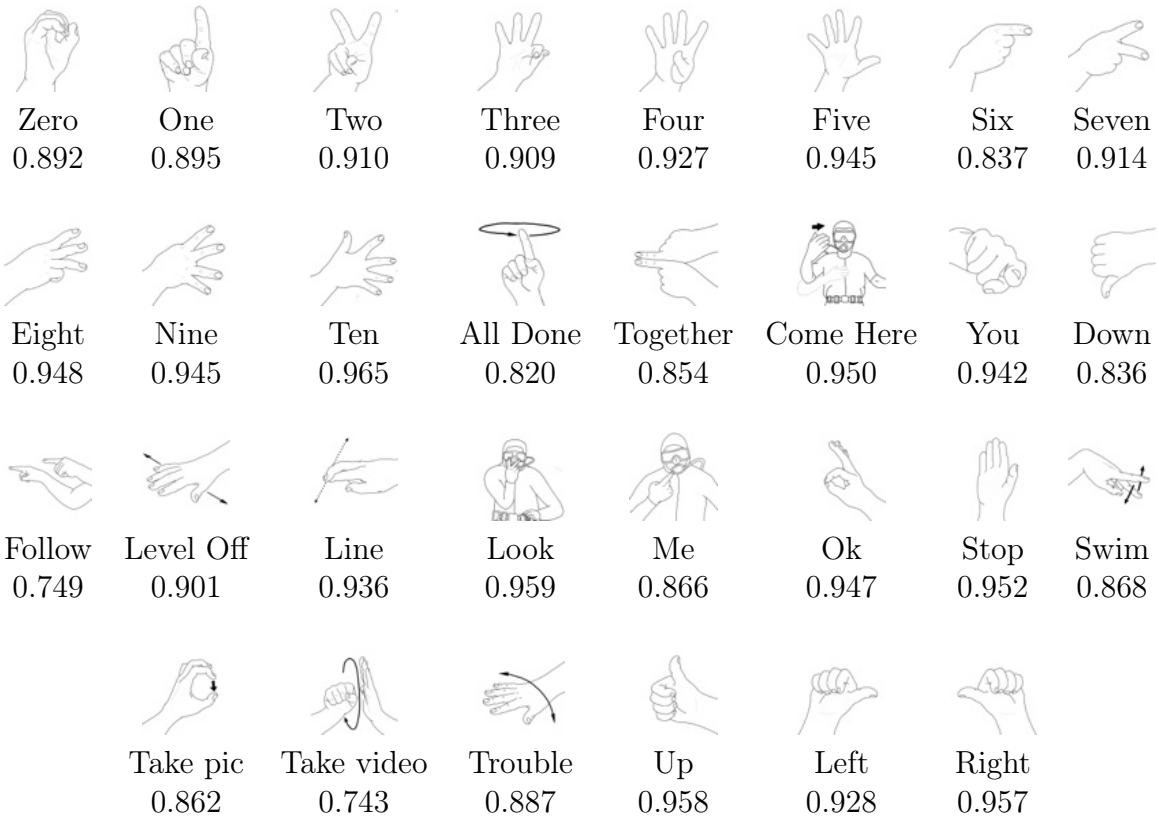


Figure 5.5: **ScubaNetV2 F1 Scores**
Shown are the F1 scores for SCUBANetV2 on each of the hand gesture classes.

Rather than being confused primarily by some potentially similar alternative gesture, the label being returned often corresponded to the gesture being demonstrated in that clip. This is illustrated in Figure 5.4(c) which plots the percentage of the mislabelled Idle frames that were labelled with the gesture being demonstrated in that data collection session. For many of the label classes more than 80% of the non-idle labels corresponded to the gesture class that was being demonstrated. A further review of the timing of these labels in a given gesture sequence showed that the network had generalized the definition of the gesture and was identifying human-labelled idle frames as expressing the gesture that just had been, or was just about to be, signalled.

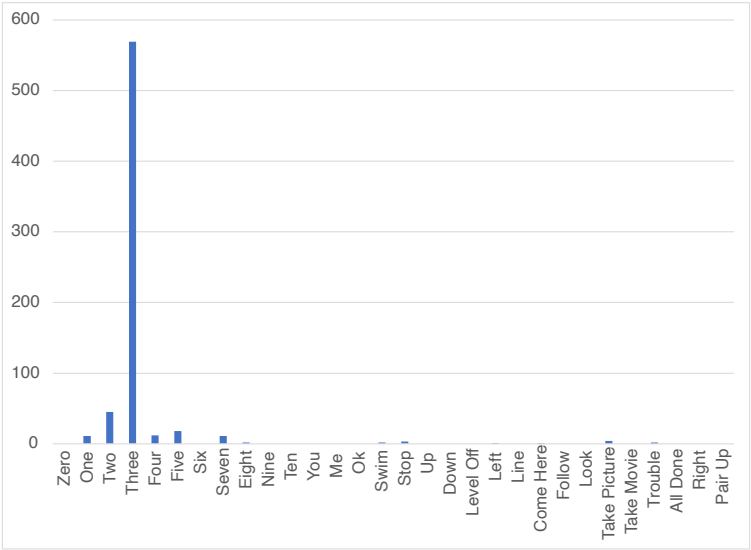


Figure 5.6: **Distribution of Idle errors.** For the gesture 'three', the maximum error class corresponds to the network identifying manually labelled Idle frames as containing the gesture for three. Many of the gestures identified correspond to gestures that appear similar to the gesture 'three'.

This is not to imply that all of the mislabelled frames were a result of the network (perhaps properly) generalizing the appearance of a given gesture. This validation also identified similarities between various gestures, especially as the diver was starting to form the gesture or to withdraw it. Figure 5.6 shows the distribution of errors associated with idle frames when the symbol for three was being presented. The symbol for three is three fingers extended vertically with the other fingers closed (see Figure 5.1). Interestingly some idle frames were confused with the one, two, four and five gestures, which present in a very similar fashion to the gesture for three as the symbol for three is formed or withdrawn.

Validation of performance using data highly correlated with the training data as done here has the potential of providing lower error rates than might be encountered with novel data. In order to address this a small set of long diver expressions (e.g., a diver communicating "You Swim Down Three Five Stop, Take Picture, Ok") were collected and run on the trained SCUBANetV2. These expressions were hand labelled with the various gestures that make

up these expressions. Table 5.4 shows the gesture sequences as text and the label and idle recognition performance percentages obtained with SCUBANetV2. A frame was identified as being Idle if one or more hands were detected and no other gesture was found in the frame, and a frame was labelled as being labelled correctly if SCUBANetV2 identified that label as the hand gesture with the highest confidence in the frame. The average %correct Idle score was 41.28% and the average %correct label score was 58.64%.

5.8 Summary

A key problem in any gesture-based language is developing a visual process that can identify the symbols that make up the language. Modern approaches to this task tend to seek to develop a machine learning-based approach that utilize some sort of supervised approach. This typically involves hand-labelling a large set of images containing the gestures that make up the language. For gesture-based languages such as ASL, this has proven to be a very difficult task as the gestures can be quite subtle, and are designed to be presented quickly and to encode a large amount of data. In contrast, application-specific languages such as SCUBA gestures are designed to be easily understood, at the cost of subtlety and speed. Even so, learning the large set of symbols required for SCUBA gestures introduces a formidable data annotation task. Here we avoid this task by leveraging a simpler labelling task, SCUBANetV1+, and then collect data in organized sessions that consist of a single gesture separated by idle segments.

This weakly-supervised approach uses a network trained on a general class (hand) and then uses this network to label specially constructed temporal sequences of individuals making a specific gesture.

Recognition rates are very high on the ScubanetV2 dataset. This is likely due, in part, to

the highly correlated nature of the dataset. Evaluation against completely novel data showed a lower recognition rate, as was to be expected, but performance rates are still above 50%. As the camera collects data at frame rates above 15hz, there will be a significant number of 'true positives' for each gesture. The process of providing feedback and/or increasing the number of true positives for a given gesture is considered in the context of gesture sequence recognition in the following chapters.

Evaluation of SCUBANetV2 shows a strong ability to identify individual gestures and to distinguish them from non-gesture hand motions. As gestures are formed and relaxed these intermediate hand positions can be confused with other gestures and this is seen in recognition in both isolated gesture examples as well as in more lengthy expressions that incorporate different gestures. Traditional temporal aggregation structures (e.g., temporal averaging using linear filters) and temporal integration using recurrent neural networks to produce clean textual descriptions of the commands being communicated by gesturing divers is presented in the next chapter.

#	Expression	%Idle	%Label
12	YOU SWIM UP NUMBER OK	79.97	65.52
8	YOU SWIM DOWN NUMBER STOP TAKEPIC OK	56.58	57.95
27	YOU OK	71.28	67.79
23	YOU FOLLOW LINE OK	61.86	48.55
27	YOU LOOK ME OK	65.37	69.22
11	YOU SWIM DOWN TAKEPIC COMEHERE OK	59.64	52.65
14	YOU SWIM DOWN NUMBER TAKEPIC COMEHERE OK	42.36	37.20
7	YOU FOLLOW ME OK	55.96	54.49
9	YOU FOLLOW ME TAKEMOVIE OK	63.92	48.08
6	YOU STOP OK	44.96	70.95
8	YOU FOLLOW ME OK	41.20	12.61
8	YOU UP OK	61.94	42.28
8	YOU DOWN OK	69.62	32.90
10	YOU ALLDONE OK	57.42	34.44
16	YOU COMEHERE OK	62.71	31.76
11	YOU LEVELOFF OK	51.56	13.43
10	YOU ME ALLDONE OK	75.31	24.72
9	YOU TAKEPIC ME OK	58.86	26.76
10	YOU LOOK UP OK	51.76	52.09
6	YOU TAKEMOVIE ME OK	66.04	51.04
10	YOU FOLLOW LINE OK	68.62	41.09
9	YOU COMEHERE LEVELOFF OK	62.47	23.45
7	YOU LOOK DOWN OK	42.22	53.75
6	YOU STOP TAKEPIC OK	40.84	58.68
10	YOU STOP TAKEMOVIE OK	66.79	42.80
7	YOU STOP COMEHERE OK	68.98	29.97
11	YOU SWIM UP NUMBER TAKEPIC COMEHERE OK	47.42	21.31
4	YOU FOLLOW TAKEMOVIE OK	46.26	65.11
9	YOU TROUBLE OK	63.07	21.13
8	YOU ME TROUBLE OK	55.89	3.59
7	YOU LOOK ME TROUBLE OK	56.97	24.45
MEANS		58.64	41.28

Table 5.4: **SCUBANetV2 Performance**
Each row describes a complex expression constructed from SCUBANETV2 gestures along with the number of recorded sequences for each expression. These expressions were captured by having a diver present continuous gesture sequences. For example, to present You, Swim, Down, Three, Five, Stop, Take Picture, OK as a continuous sequence. For each expression the percentage of idle frames and percentage of non-idle frames that were correctly labelled are reported. The average percentage across expressions of idle frames and non-idle frames is reported.

Chapter 6

Language Translation

6.1 Introduction

The model described in the previous chapter can reliably detect the SCUBA gestures shown in Figure 1.4 when they are presented in isolation. Performance when they are presented as part of a larger expression demonstrated two issues. First, the transition periods between gestures can be confused with other gestures, and "Idle" gestures can be incorrectly recognized as a non-Idle gesture. As a consequence, the output of the model over the length of a complex gesture expression is a noisy sequence of a set of detected objects each with an associated gesture class, confidence value, and bounding box. This is illustrated in Figure 6.1 which shows a sequence of frames from the sequence "You Swim Up Six Ok". The approximate number of frames for each gesture, is shown under each frame. Each frame in the sequence was run through the SCUBANetV2 network and the gesture with the highest score is plotted by frame number. In Figure 6.1 this is plotted including idle (lower left) and excluding regions in which idle was the gesture with the highest certainty (lower right). In presenting a sequence a diver may provide ambiguous gestures, or even incorrect gestures, and when the

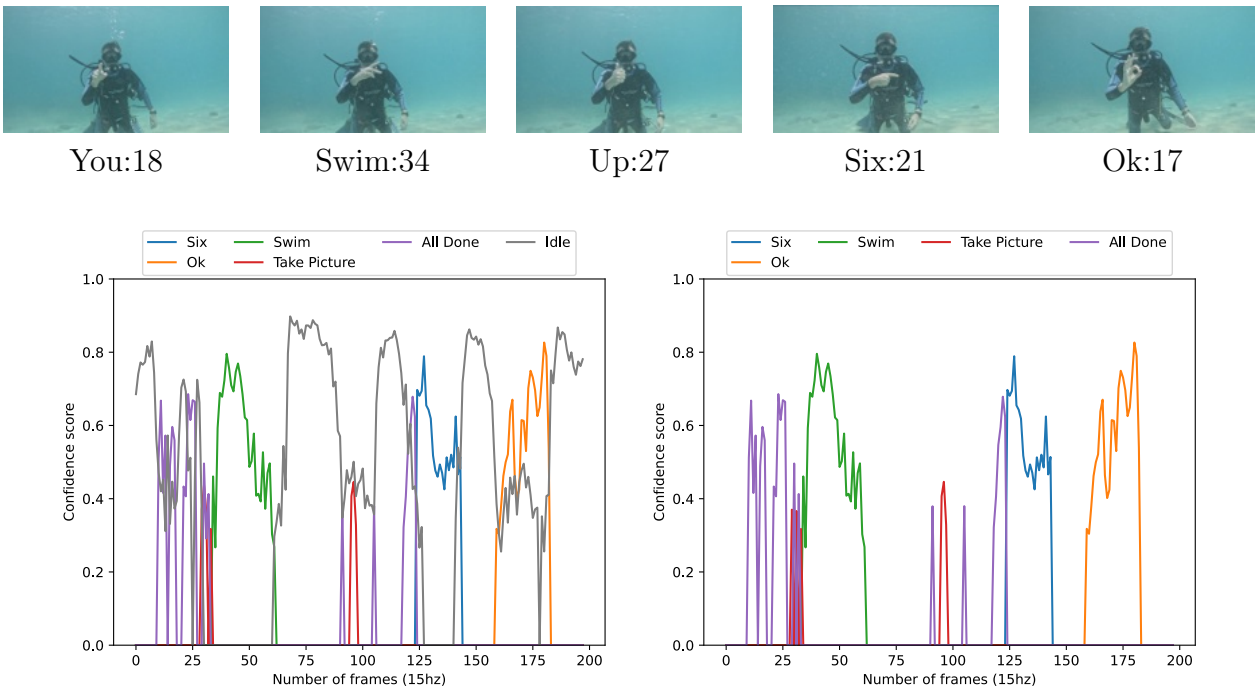


Figure 6.1: **Sample Gesture Expression Recognition**
Upper part of the image shows frames from a sequence in which the diver signaled "You Swim Up Six Ok" showing approximate number of frames in each gesture. Each gesture is separated by 'idle' frames in which the diver does not provide a signal but rather moves their hand to the next gesture, typically through some non-gesture state to provide a separation of gestures (so that the same gesture can appear multiple times in sequence). Frame counts for each gesture are shown. The plots in the lower part of the figure show the confidence scores for the gesture detected in a given frame. The plot on the left includes confidence score for all gestures detected in each frame including 'idle'. The plot on the right excludes 'idle' labels and all but the gesture with the highest confidence score in each frame.

diver is presenting the correct gesture the recognizer may not recognize the gesture correctly.

Determining the true intent of the sequence of gestures can be conceptualized as a language translation problem; taking the individual frame recognition confidences and mapping this noisy signal to the textual labels. Training a model to understand such sequences can be expressed as a supervised training problem within which human experts in the gesture language label a large corpus of data. This would be an extremely difficult and time consuming task as a data gesture sequence can take 30-60 seconds to present (and likely much longer to

label) and thousands of such gesture sequences would be required. Given the difficulty of obtaining such human-labelled data, here I follow an approach in which weakly supervised data within a sim2real process is used to train a LSTM-based network to recognize common underwater gesture sequences. Training takes place using an artificially constructed set of underwater gesture sequences. Evaluation is performed using the Sim2Real dataset as well as using hand-labelled data collected at depth.

6.2 Background

Recurrent neural networks (RNNs) such as long short term memory (LSTM) have found wide application in the recognition of such gestures terrestrially (see [108] for a recent example). A critical requirement of this type of approach is a large annotated dataset of gesture sequences to train the RNN. Unfortunately, curating a large annotated dataset is a difficult task, requiring humans with expertise in the gesture language to label the datasets.

One approach to obtaining such a dataset without manually labelling a large data corpus is through simulation (see [26] for example). Although this addresses the issue of insuring a proper correlation between the script and the resulting data sequence, it introduces the issue of ensuring that the simulation properly captures reality. A number of approaches exist in the literature to address the "sim2real gap", including addressing it in system training (e.g., by integrating both training on simulated data as well as manually labelled data as in [97]) or utilizing a post-video generation process to reduce the sim2real gap as in [45]). Weak supervision provides an alternative approach to the creation of fully labelled datasets. A number of different weak supervision approaches are identified in the literature. Many of these approaches are intended for static image labelling. For example, [42] describes the application of weak supervision in Covid 19 CT analysis. See [135] and [94] for a brief

review of the approach. In the action segmentation space, one weakly supervised approach that has shown considerable promise is the use of action segmentation through the use of a transcript (see [68], for an example). Given a video sequence with a known transcript, properties of the transcript can be used to segment the sequence where each sub-sequence corresponds to a given action. This then provides a labelled set of video frames, assuming that the temporal segmentation process is robust. Perhaps the most robust segmentation process is to use controlled data collection (i.e., to use a video that was created following a structured transcript) and manual identification of video segments. Chapter 5 describes a weakly-supervised approach to labelling specific frames of diver gestures and then constructs a network using Yolo [98] to recognize the diver gestures shown in Figure 5.1. The resulting network can be used to recognize specific gestures but it does not recognize specific expressions in terms of their role within a sequence of gestures forming a specific command. Here a combination of a weak supervision technique and a synthetic action video constructed from structured data collection is used to address the problem of translating gesture expressions to natural language, suitable for commanding an underwater robot.

6.3 Data Collection

As shown in Chapter 5, SCUBANetV2 can be used to generate an agent capable of recognizing individual SCUBA gestures. The dataset can also be used to assemble synthetic gesture sequences following an approach inspired by [105]. The SCUBANetV2 data collection process obtained a number of video sequences of divers presenting a specific gesture separated by "Idle" gestures. This enables the assembly of video sequences of divers communicating specific gestures sequences from real data captured under realistic conditions. Given the nature of the dataset it is possible to assemble the synthetic gesture sequences at different levels

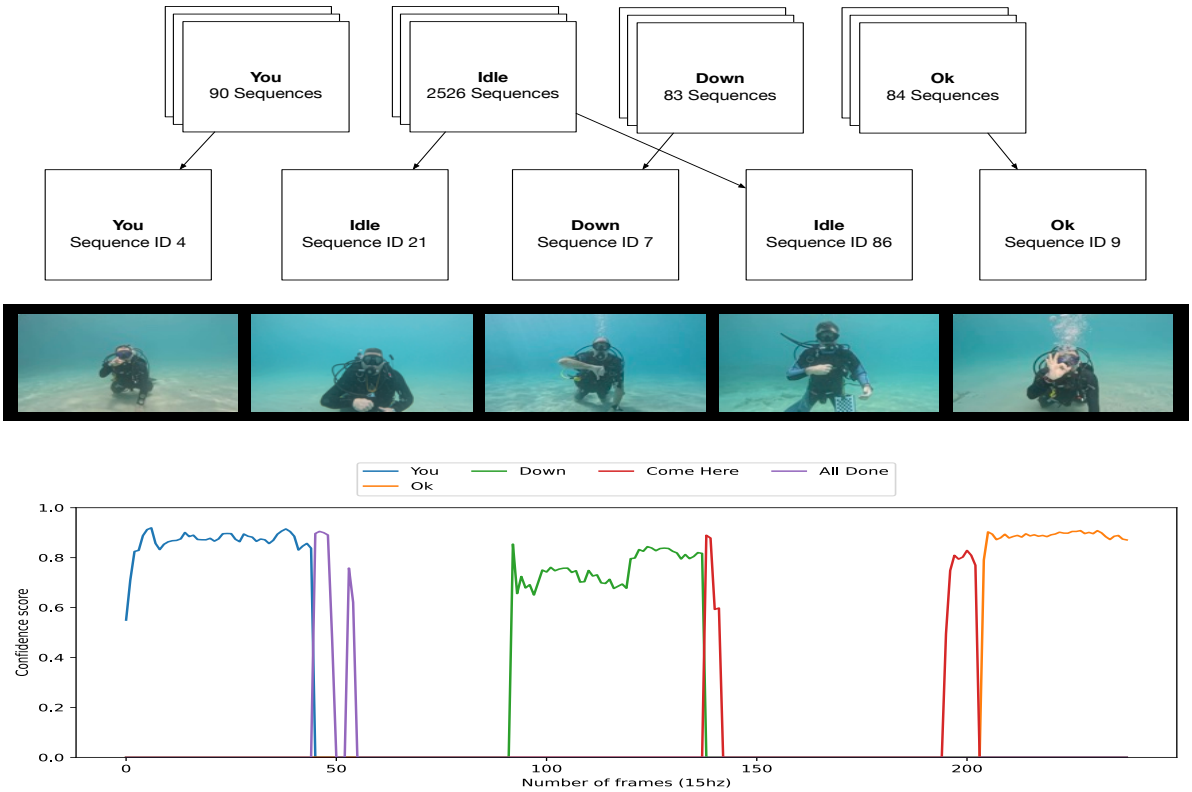


Figure 6.2: **Assembling a Synthetic Gesture**
Individual gesture examples, such as the one for "Up" shown in Figure 5.2 are assembled into a synthetic gesture sequence. An expression sequence is assembled from randomly chosen examples of each of the gesture sequences that make up the expression, separated by randomly chosen idle sequences. Although the simulated video stream shows abrupt visual changes, the actual SCUBANetV2 signal shown in the lower portion of the figure captures the dynamics of each gesture being presented, including idle gestures. It also captures the temporal distribution of individual gestures (how long the diver takes to form, present and then withdraw a specific gesture) and idle periods. This synthetic image sequence also captures the nature of the presentation and withdrawal of a gesture which, on occasion, can be easily confused with other gestures in the gesture set. Here for example, the withdrawal of the Ok gesture was recognized as an intermittent "All Done" gesture.

of granularity; that is, from using the individual frames to the entire symbol presentation sequences as elements of the simulation. Frames or segments could be drawn from either one individual diver or from the full set of divers associated with the dataset or from some subset. Here the entire individual labelled gesture sequence was chosen, and selected segments across

all divers in the dataset. These larger segments were chosen for three reasons. (i) Following the advice given in [105] "the minuteness of the parts formed a great hindrance to my speed", the use of longer sequences enables a more speedy assembly of the whole. (ii) By utilizing the entire gesture presentation, temporal properties associated with forming and releasing the gesture and idle periods are captured. And (iii), integrating across divers enables the assembly of an extremely large set of SCUBA gesture sequences.

Each assembled SCUBA gesture sequence results in a spatio-temporal gesture sequence as illustrated in the top portion of Figure 6.2. A sequence is selected by assembling a randomly chosen presentation of each of the gestures that make up the expression separated by randomly chosen idle sequences. Although the resulting sequence can show a significant discontinuity between the "stitched together" video segments, when represented in a head-centred reference frame, the resulting sequence of gestures and bounding boxes is representative of a diver signalling this sequence. This is illustrated in the lower portion of the figure which shows the SCUBANetV2 encoding of the sequence. Observe that by utilizing full sequences, and not just utilizing a synthetic sequence of labels, a closer approximate of what the image processing pipeline will experience when operating on non-synthetic data sequences can be achieved.

SCUBA communication at depth is typically bookended by standard gestures, e.g., "You" to start a communication and "Ok" to signal the end. That standard is followed here. A language of 64 likely expressions (see Figure. 6.3) was used to generate 706 sample expressions. Each expression is of the form

<gesture 1> <idle> <gesture 2> <idle> ..... <idle> <gesture n>

where <gesture 1> is a randomly chosen gesture sequence for "You" and <gesture n> is a randomly chosen sequence for "Ok". Each of <gesture 2> ... <gesture n-1> were drawn from

1. You Swim DIRECTION NUMBER Ok
2. You Swim DIRECTION NUMBER Stop TakePicture Ok
3. You Ok
4. You Follow Line Ok
5. You Look Me Ok
6. You Swim DIRECTION TakePicture ComeHere Ok
7. You Swim DIRECTION NUMBER TakePicture ComeHere Ok
8. You Follow Me Ok
9. You Follow Me TakeMovie Ok
10. You Stop Ok
11. You TakePicture Ok
12. You DIRECTION Ok
13. You AllDone Ok
14. You ComeHere Ok
15. You LevelOff Ok
16. You Me AllDone Ok
17. You TakePicture Me Ok
18. You Look DIRECTION Ok
19. You TakeMovie Me Ok
20. You ComeHere LevelOff Ok
21. You Stop TakePicture Ok
22. You Stop TakeMovie Ok
23. You Stop ComeHere Ok
24. You Swim DIRECTION NUMBER TakePicture ComeHere Ok
25. You Follow TakeMovie Ok
26. You Trouble Ok
27. You Me Trouble Ok
28. You Look Me Trouble Ok
29. You Swim DIRECTION Ok
30. You Look DIRECTION TakePicture Ok
31. You Swim DIRECTION NUMBER Stop Ok
32. You Swim DIRECTION NUMBER TakePicture Ok
33. You Swim Line NUMBER TakePicture Ok
34. You Follow Line NUMBER TakeMovie Ok
35. You Stop Look Trouble Ok
36. You Follow Line DIRECTION AllDone Ok
37. You Stop Look DIRECTION Ok
38. You Stop Look Direction AllDone Ok
39. You Stop Follow Line NUMBER Stop Ok
40. You Follow Line NUMBER Ok
41. You Stop LevelOff ComeHere Ok
42. You Together Me Ok
43. You Follow Line DIRECTION NUMBER Ok
44. You Swim DIRECTION NUMBER Ok
45. You Swim DIRECTION NUMBER ComeHere Ok
46. You Swim NUMBER DIRECTION TakePicture Ok
47. You TakeMovie Follow Line Ok
48. You TakeMovie Swim NUMBER DIRECTION Ok
49. You Swim DIRECTION NUMBER TakePicture ComeHere Ok
50. You ComeHere TakePicture Swim NUMBER Stop Ok
51. You Stop AllDone Ok
52. You Look DIRECTION Trouble Ok
53. You Stop Look DIRECTION Trouble Ok
54. You Stop Look DIRECTION Swim NUMBER TakePicture Stop Ok
55. You Follow Line Look Trouble Ok
56. You Follow Line DIRECTION Look Trouble TakePicture ComeHere Ok
57. You Follow Line DIRECTION TakeMovie Look Trouble ComeHere Ok
58. You Trouble DIRECTION TakePicture Ok
59. You Trouble DIRECTION TakeMovie Ok
60. You Stop DIRECTION TakeMovie Ok
61. You TakeMovie DIRECTION Ok
62. You Swim Me Stop AllDown DIRECTION Ok
63. You ComeHere AllDone Up Ok
64. You ComeHere Stop DIRECTION TakeMovie NUMBER Stop Ok

Figure 6.3: **SCUBALang Gesture Expressions**
Capitalized words NUMBER and DIRECTION are meta-gestures used to further expand this list. An expression with NUMBER is duplicated 10 times and replaced with a number between 0-999. An expression with DIRECTION is duplicated four times and replaced with either (UP, DOWN, LEFT, RIGHT).

the random sequences for the appropriate symbol in the expression being generated. This randomized process was repeated 250 times for each expression. For expressions involving numbers, a random number in the range of 0 through 999 lacking proceedings zero's was used, and for directions, a meta gesture for <direction> was used instead, which was replaced with

Template expressions	64
Unique expressions	706
Total expressions	176,500
Avg # of gestures/expression	7.5
Avg # of frames/expression	840
Total hours	2745.5

Table 6.1: **SCUBALang Statistics**
In the SCUBALang dataset there are 64 template expression which are expanded into 706 unique expressions. Each unique expression is assembled 250 times from randomly selected clips of gestures, resulting in 176,500 gesture expression video sequences totaling 2745.5 hours of video. There is an average number of 7.5 gestures and 840 frames per expression

a gesture for a random direction chosen from "Up", "Down", "Left", or "Right". This weakly supervised and artificially assembled dataset of 176,500 sequences was then used to train a recognizer for SCUBA expressions. Table 6.1 details some of the key statistical attributes of the SCUBALang dataset. Figure 6.2 summarizes the process of obtaining a synthetic expression.

6.4 Data Annotation

Each assembled expression sequence is processed by SCUBANetV2 to obtain a sequence of labelled bounding boxes. If multiple non-idle gesture labels are encountered in a given frame, only the one with the highest certainty is retained. This produces a long sequence of gestures representing the dominant label for each frame in the assembled video. This is illustrated in the plots associated with the sequences shown in Figure 6.1 and elsewhere in this chapter. Rather than retaining the raw encoding as is, a run length encoding scheme is used to provide a more compact representation of these sequences. The length of each run is represented by the integer value of the $\lfloor log_2 \rfloor$ of the run length. This reduces the size of the vocabulary of the resulting sequence. This process is illustrated in Figure 6.4, where the output is You [2]

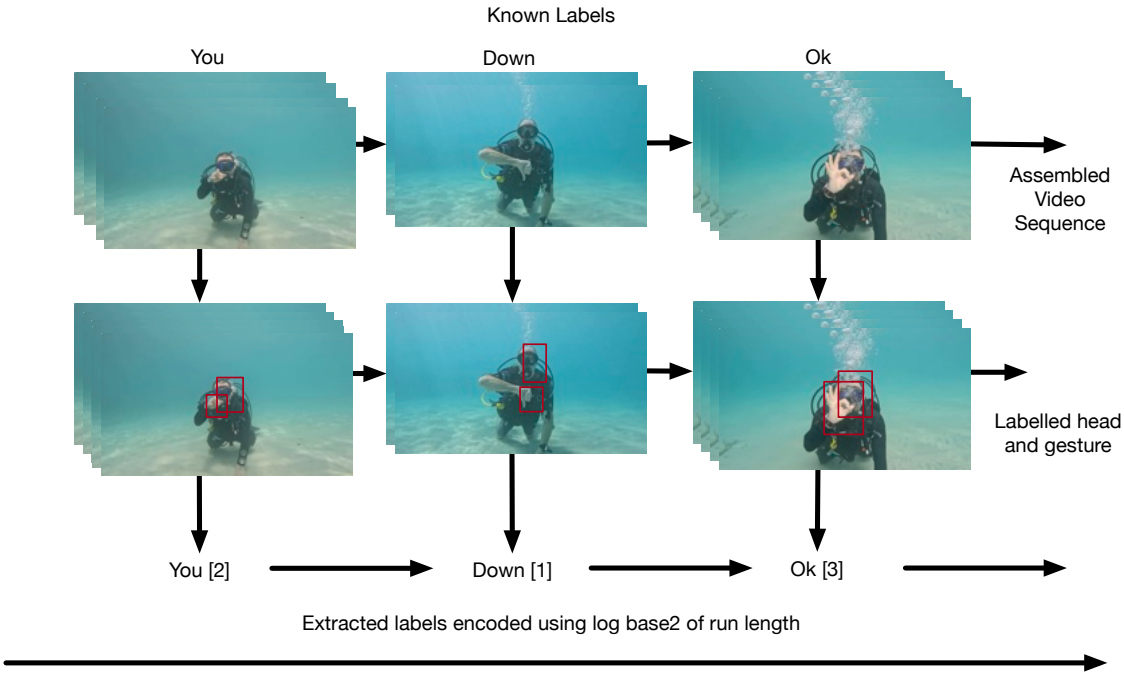


Figure 6.4: **Temporal Annotation of a Synthetic Gesture Sequence** Each video sequence is processed by SCUBANetV2 to obtain a dominant gesture for each frame. The dominant gesture defaults to "Idle" if "OffHand" or no gesture is found. The resulting sequence of dominant gestures is run length encoded where the run length is replaced with the $\lfloor log_2 \rfloor$ of its value.

Down [1] Ok[3].

6.5 Model Training

A straightforward LSTM network was used to map gesture sequences to robot commands. The architecture of the network is sketched in Figure 6.5. An input vocabulary was constructed from the set of words that can make up an input expressions, which consists of all gestures (including IDLE), the log2 encoding lengths [1] - [9] and two special words ("", [UNK]). The symbol "" is used to represent padding and [UNK] is used to represent unknown words. Padding is used to ensure all inputs into the LSTM are the same length. Similarly, a target vocabulary was constructed from the set of words that can make up a target expression,

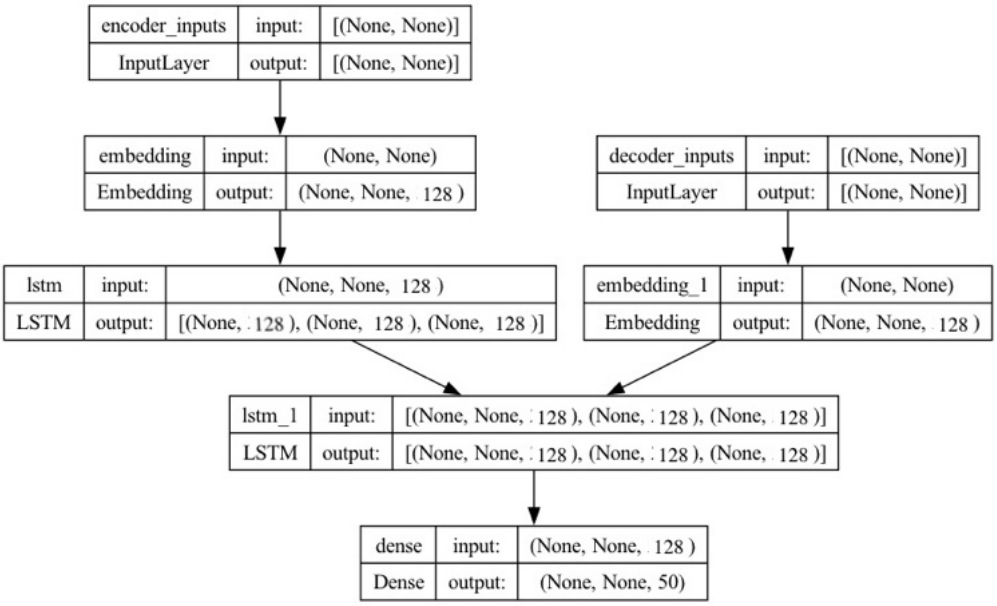


Figure 6.5: **LSTM Network architecture.**
Layer types and sizes of the LSTM-based Seq2Seq Encoder-Decoder model.

which consists of all gestures (excluding "Idle") and two special words ("", [UNK]). Each input and target expression in the dataset was converted to integer sequences representing their index in a their respective vocabularies. The dataset was split into train/validate/test using a ratio of 70/15/15. A basic LSTM-based Seq2Seq encoder-decoder model was used following the model described in [110], with an embedding layer preceding the LSTM layer in both the encoder and decoder and with an output dimensionality of 50. This limits the length of gesture sentences to 50 gestures. This should be sufficient for the task here. The model is trained using a sparse categorical cross-entropy loss function over 30 epochs achieving an accuracy of 98.5% on the test set.

6.6 Validation

A common metric for evaluating the performance of machine translations is the BLEU (BiLingual Evaluation Understudy) score[85]. The BLEU score represents the similarity of the machine-translated text to a set of high quality reference translations. BLEU scores can be measured on unigrams, bigrams, trigrams, and four-grams, with unigrams measuring adequacy and four-grams measuring fluency. Gram scores ≥ 60 are considered high-quality at or above human capabilities[85].

The uni-gram and bi-gram BLEU score of the models translations were calculated across three different testing sets shown in Figure 6.6. The tri-gram and four-gram BLEU scores were omitted because a large number of collected expressions were not long enough for these scores to be effective metrics. The first set, with results shown in Figure 6.6 set (a) is comprised of 1000 randomly selected artificial sequences from the synthetic testing set initially used to evaluate the accuracy of the model. The second set with results shown in Figure 6.6 set (b) consists of 120 hand-labelled videos of divers performing one of 25 different expressions. Average BLEU scores are given in Table 6.3.

Performance in terms of BLEU scores is very good for the synethetic dataset in terms of both uni-gram and bi-gram BLEU scores. Although mean BLEU score performance was still good for the hand labelled sequences, a number of expression sequences had very poor scores. An examination of the labelling of these sequences shows that a number of these poorly performing sequences were a result of the individual gestures that made up the sequences not being recognized at any point in the expression and thus not included in the encoding passed to the LSTM. A review of the raw video showed that although the individual gestures are easily identified, the divers presenting the gestures could present certain gestures "very briefly", and that this might be so quick as to not provide a clear example of the individual

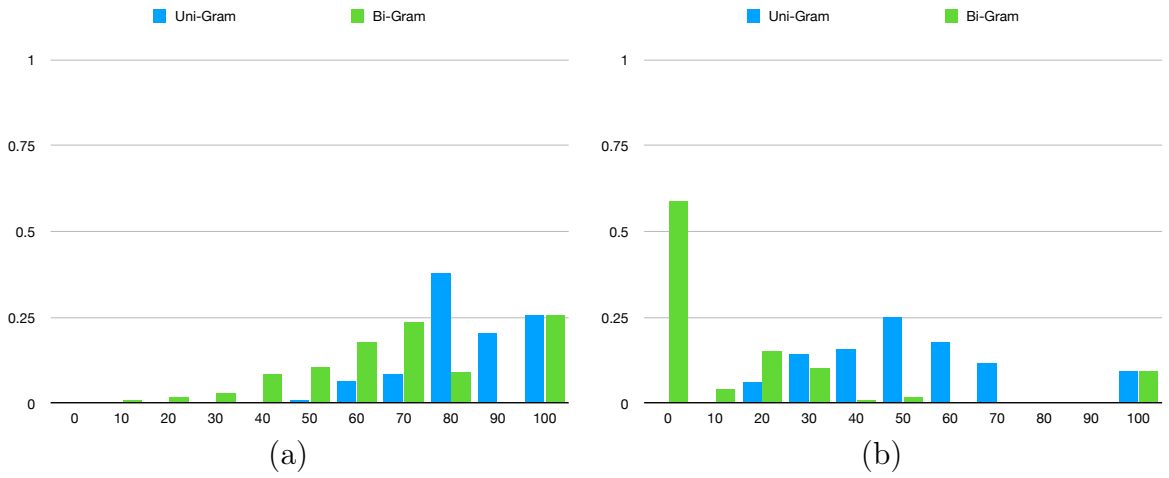


Figure 6.6: **BLEU Scores**
Uni-Gram and Bi-Gram scores report across three separate sets of expression data. (a) BLEU Scores performance on 1000 randomly selected artificial sequences. (b) shows performance on 120 hand-labelled videos of divers performing one of 25 different expressions.

gesture to the robot. Figure 6.7 plots the maximal SCUBANetV2 output for each frame in the sequence YOU SWIM LEFT 10 OK. In other circumstances divers presented a gesture that was difficult to determine from the camera view point or the gesture was presented in a non standard way, resulting in the gesture not being detected.

6.7 Improving Gesture Recognition

The less than ideal recognition performance of the LSTM for quickly/poorly presented gesture sequences is not a surprise. An LSTM requires coherent data that matches the data on which it was trained. If the network is unable to recognize many (or any) examples of a critical signal in the gesture sequence then performance is likely to be poor. Errors in gesture sequence recognition suggested that in terms of SCUBA gesture recognition two factors were at play (i) the key gestures in a given sequence had to be recognized, and they had to be recognized for a sufficiently long period of time to be consistent with the temporal nature

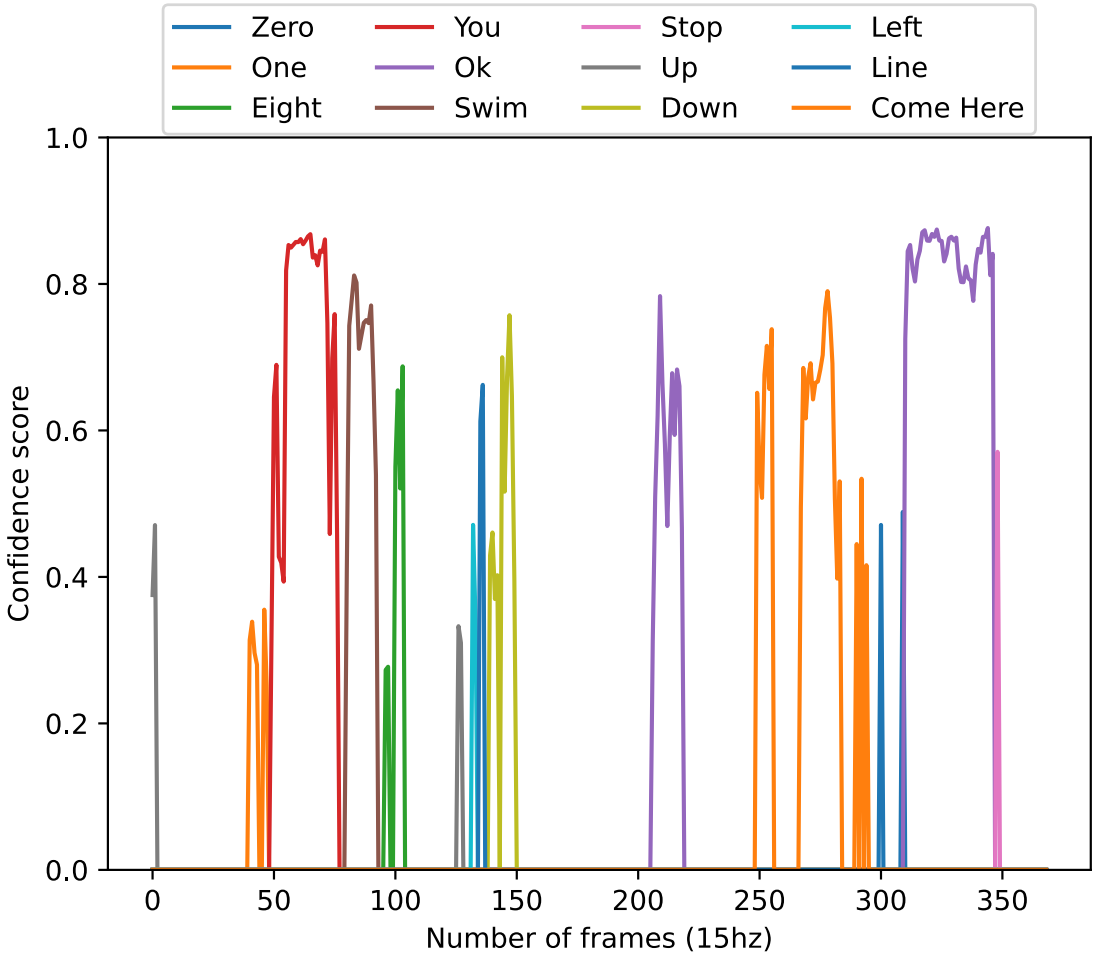


Figure 6.7: **Poor Recognition Example**
SCUBANetV2 maximal responses to the input gesture sequence You Swim Left 10 Ok.

of the training data. The Sim2Real training data forced the user to present features clearly and for a reasonable duration. These assumptions were validated in two ways. First, by evaluating the performance on the LSTM on data for which the required gestures had been recovered, and second by ensuring that the diver presented data sequences sufficiently slowly to provide the LSTM with sufficient gesture data to recognize the gesture sequence.

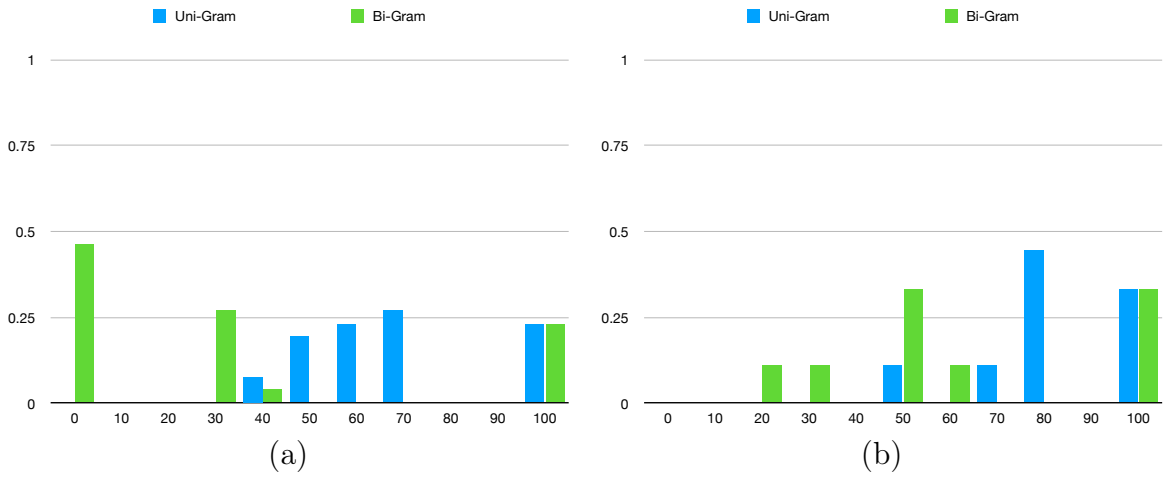


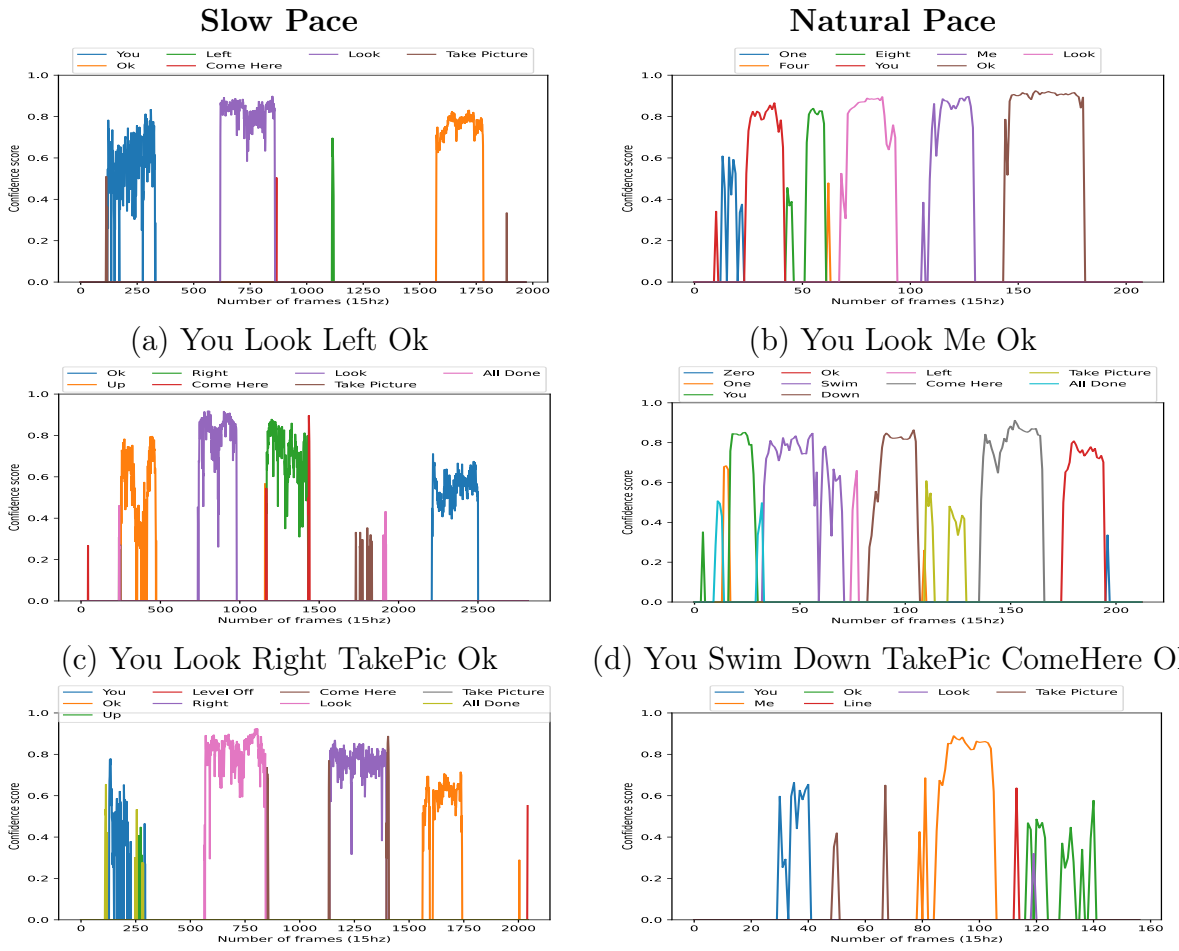
Figure 6.8: **BLEU Scores II**
Uni-Gram and Bi-Gram scores report across three separate sets of expression data. (a) shows performance on 25 labelled videos drawn from the previous real world testing set where the SCUBANetV2 model correctly identified each gesture in the target expression in at least 10 frames. (b) shows performance on 10 expressions presented slowly to the robot.

6.7.1 Requiring a Minimum Number of Gestures in the Sequence

To explore the first assumption, the hand labelled test dataset was trimmed of any gesture sequence for which each required gesture in the sequence was not recognized in at least 10 frames. This reduced the total number of gesture sequences to 25 (a 92% reduction). This improved performance considerably. Figure 6.8(a) shows the minimum gesture presentation BLEU performance. Not surprisingly, requiring that the diver present the gestures that are required to make up the gesture sequence improves recognition performance.

6.7.2 Slowing Down Gesture Presentation

The assumption that slowing down gesture presentation to obtain a more robust set of individual gestures would improve recognition performance was tested by collecting a small number of test phrases where a diver was asked to present gestures for a count of 5 and returning to an idle state for a count of 5 in between each gesture. This resulted in half of



(a) You Look Left Ok

(b) You Look Me Ok

(c) You Look Right TakePic Ok

(d) You Swim Down TakePic ComeHere Ok

(e) You Swim Right Ok

(f) You Look Up Ok

Figure 6.9: **Slow vs Natural Gesture Presentation**
(a,c,e) time series graphs of the confidence values of all non-idle gestures detected by SCUBANetV2 in gesture expressions where each gesture is held for approximately 5 seconds. (b,d,f) time series graphs of the confidence values of all non-idle gestures detected by SCUBANetV2 in naturally paced gesture expressions. Slowing down gesture presentation improves the signal to noise ratio but is not immune to missed or false detections.

the test phrases being successfully detected. The elements in this test dataset are shown in Table 6.2. Figure 6.9 provides an example of how slowing down diver presentation changes the way in which the gestures are presented and their recognition. Shown here are three gesture sequences from the hand labelled test dataset and three from the longer presentation dataset. BLEU performance on this dataset is shown in Figure 6.8(b), and BLEU performance is

Expression	Detected Expression
You Look Left Ok	You Look Left Ok
You Look Left Ok	You Look Left Ok
You Look Left TakePic AllDone Ok	You Look Left TakePic Ok
You Up Look Right TakePic AllDone Ok	You Look Right TakePic Ok
You Swim Left Stop Ok	You Swim Left Ok
You Swim TakePic Swim Down Ok	You Swim Left Ok
You Swim Left Stop Ok	You Swim Left Ok
You Swim Right Ok	You Swim Right Ok
You AllDone Swim Right Ok	You Swim Right Ok

Table 6.2: **Slowed Gesture Expression**
Lists the set of test expressions in the slowed dataset as well as the expression detected by the SCUBALang LSTM.

given in Table 6.3. Ensuring that the diver presents gestures clearly leads to a much more successful gesture sequence recognition system.

The majority of failed tests with the slow presentation dataset were a result of a single gesture not being detected as the gesture did not align with SCUBANetV2's internal representation. While the slow strategy does result in an improved detection rate it dramatically slows down the rate at which the diver can communicate to the robot. The average length of a gesture is 21 frames (1.4 seconds) and 14 frames (0.93 seconds) for an idle gesture. A naturally paced expression sequence of five gestures takes on average 10.7 seconds, using this delay tactic extends the presentation window to 45 seconds. This extra time is not necessarily the same as making sure that the figure is properly extracted. This is illustrated in Figure 6.9a where the LEFT gesture is only recognized in a very few number of frames even with a slow presentation of the sequence. Ensuring that the diver presents their command sequence in a *slow* and *clear* manner is critical for effective recognition. This can be accomplished by artificially lengthening the time each gesture is presented, but this introduces artificial delays in terms of gesture sequence presentation nor does it always guarantee clear presentation. What is really needed is a feedback mechanism to provide to the user a cue that their current

	Uni-gram	Bi-gram
Set (a) Synthetic	83.3	67.0
Set (b) Hand-labelled	52.5	16.8
Set (c) Reduced Hand-labelled	68.4	32.6
Set (d) Slowed presentation	0.82	0.62

Table 6.3: **Mean Uni-gram and Bi-gram Scores**

gesture has been properly received. This involves embedding the recognition process within a bi-directional communication. The next chapter explores feedback from the detection system to the diver to improve communication with the robot.

6.8 Summary

The Sim2Real approach described here that leverages the incredibly large dataset made by possible due the weakly supervised approach used in SCUBANetV2 results in a system that performs remarkably well on simulated data. It also performs well on natural data sequences, but at a rate that would likely be too poor to be used *as is* in a commercial setting. An investigation of the cause of the poorer performance suggests that ensuring that divers present gestures clearly and consistently over a number of frames is likely to address this issue.

Given the significant improvement in BLEU scores when ensuring gestures are presented for at least a given amount of time it stands to reason that if the robot was to show the diver the gesture that it was recognizing at a given time – by presenting the output of the SCUBANetV2 model at each time instance, for example – the diver could modify their presentation of the gesture for better clarity and to be informed that they can move on to the next gesture in the communication sequence. Another strategy is to have the robot replay the interpreted expression sequence to the diver and not execute the command until the diver confirms that the robot's understanding of the sequence was correct. The Milton robot[22]

shown in Figure 3.4 is equipped with a forward-facing display panel to enable exactly these positive responses to the diver companion. The next chapter describes the implementation of feedback mechanisms that assist divers in improving communication with the robot and ensures the robot only executes properly understood commands.

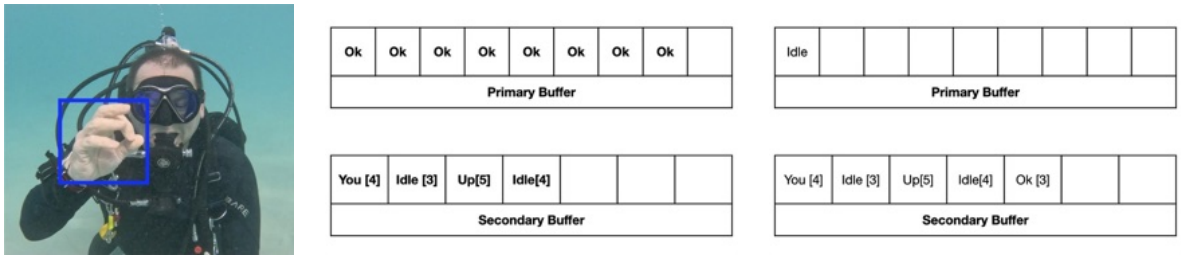
Chapter 7

Interactive Diver → Robot Communication

7.1 Introduction

7.2 Model Optimization

The work in the previous chapters utilize YOLOV8 to drive the SCUBANetV2 per-frame gesture detection process. The Ultralytics library for YOLOV8 uses the pytorch model format to serialize the neural network models it obtains. This format is convenient for training and debugging purposes as pytorch is compatible with multiple operating systems including Linux, Windows and MacOS, and various processor architectures including x86_64 and arch64. However, when constrained by the processing power of smaller GPUs such as the NVIDIA Orin Nano onboard the Milton UUV even the compact YOLOV8n and YOLOV8s models are unable to reach the desired frame rate of 15hz. To deploy this model effectively on the Nvidia Jetson platform used by Milton the network architecture needs to be converted to a



(a) (b) (c)

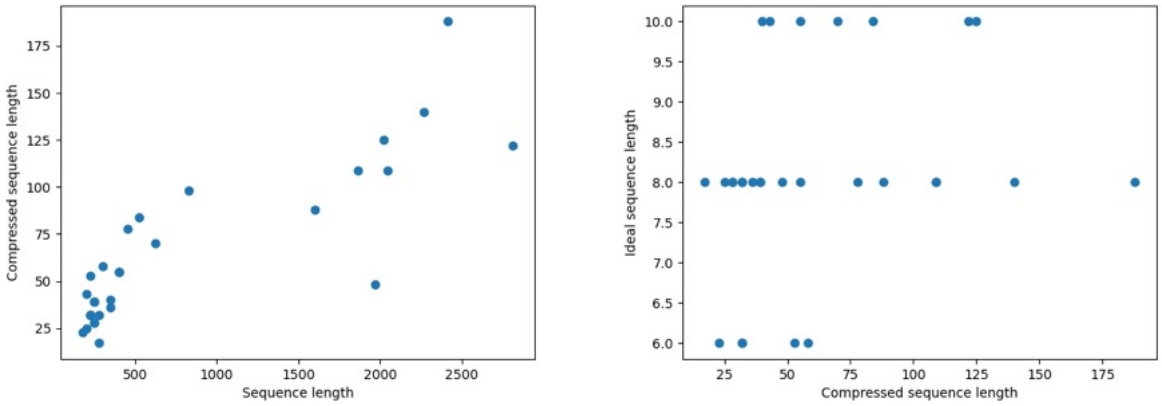
Figure 7.1: **Sequence Buffering**

Labelled gestures are processed as they arrive and are buffered. Frame labels are first added to the end of the primary buffer. (a) Incoming "Idle" gesture. (b) Shows the state of the primary and secondary buffers before processing the incoming "Idle" gesture. The incoming gesture is not the same as those stored in the primary, so its contents are run-length encoded - Ok [3] - and appended to the secondary buffer. The primary buffer is then emptied to make room for the new gesture "Idle". (c) Shows the state of the primary and secondary buffers after process the incoming "Idle" gesture.

representation better suited for the resource available on this computational platform. For onboard computation on Milton a TensorRT engine[112] is used. TensorRT is an SDK for high-performance deep-learning inference optimization and runtime built on Nvidia's CUDA library. Nvidia provides the trtexec tool which converts models defined in the Open Neural Network Exchange (ONNX) format to optimized TensorRT engine files. The ONNX initiative targets interoperability and shared optimization and most importantly is a model serialization format supported by pytorch and Ultralytics. The optimized YOLOV8s TensorRT model has a maximum inference latency of 22.188 ms – which equates to approximately 45hz – more than enough to accommodate Milton's intended camera capture rate of 15hz.

7.3 Sequence Buffering

The SCUBANetV2 YOLOV8 object detection model processes a single image at a time. The model can be run continuously on images from a live camera feed to build a long sequence of detected gestures. However, these extremely long gesture sequences are unnecessary for many real world applications including the application here. In the majority of use cases, divers will only communicate with the robot in short bursts before leaving it to execute the desired task. This property can be leveraged to optimize the sequence recognition process. In the implementation described in Chapter 6, the length of training sequences was limited to 250 symbols. Arbitrarily long gesture sequences consume a significant amount of GPU memory. In practice only a small portion of these sequences contain relevant diver communication. The remainder represents non-communication sequences, empty frames etc. To avoid decoding non-communication sequences and to reduce the rate at which sequences are fed into the LSTM encoding of SCUBALang the unencoded and encoded sequences are held in buffers which are emptied based on the LSTM output. The primary buffer holds an unencoded sequence of the dominant hand gestures detected by the SCUBANetV2 YOLO model. The primary buffer is run-length encoded, emptied and, appended to the secondary buffer when an incoming gesture is different from the previous gesture in the primary buffer. This process is depicted in Figure 7.1. The secondary buffer holds a run-length encoded sequence with a maximum size of 250 symbols to correspond with the limit imposed on training sequences. The length being encoded here is $\lfloor log_2 L \rfloor$ where L in the number of identical symbols in the run. The benefit of this encoding process can be seen in Figure 7.2 which shows the relative sequence length compression of real-world test phrases. In practice this technique can achieve an average compression ratio of 7:1. This translates to a buffer length of 250 being able to hold 1750 (250 * 7) frames worth of gestures, effectively limiting the length of phrases to



(a) Uncompressed vs Compressed (b) Actual vs Ideal

Figure 7.2: **Gesture Duration Comparison**
(a) shows the size of run-length encoded sequences fed into the LSTM as a function of their uncompressed length (number of frames). (b) shows the ideal length of compressed sequences resulting for perfect gestures detection compared with their actual compressed length. Detection errors disrupt the run length of gestures causing the actual compressed sequence length to be longer than the ideal length.

approximately 115 seconds.

If appending to the secondary buffer would cause the buffer to increase its size beyond 250 symbols then the oldest gesture in the buffer is removed. Whenever an "Ok" gesture is appended to the secondary buffer the sequence is fed into the LSTM. If the LSTM returns a completed phrase ending with the special token "[end]" and the phrase does not contain the special token "UKN" the buffer is emptied and the phrase becomes the output of the system. All other phrases returned from the LSTM are treated as invalid communication. Figure 7.1 illustrates this process further, Figure 7.1(a) shows the newly detected gesture "Idle" and Figure 7.1(b) shows the current state of the primary and secondary buffer. As the new gesture "Idle" differs from those in the primary buffer its contents are run-length encoded - resulting in the symbols "Ok" and "[3]" – and appended to the secondary buffer, the primary buffer is then emptied and the new gesture "Idle" is added the primary buffer.

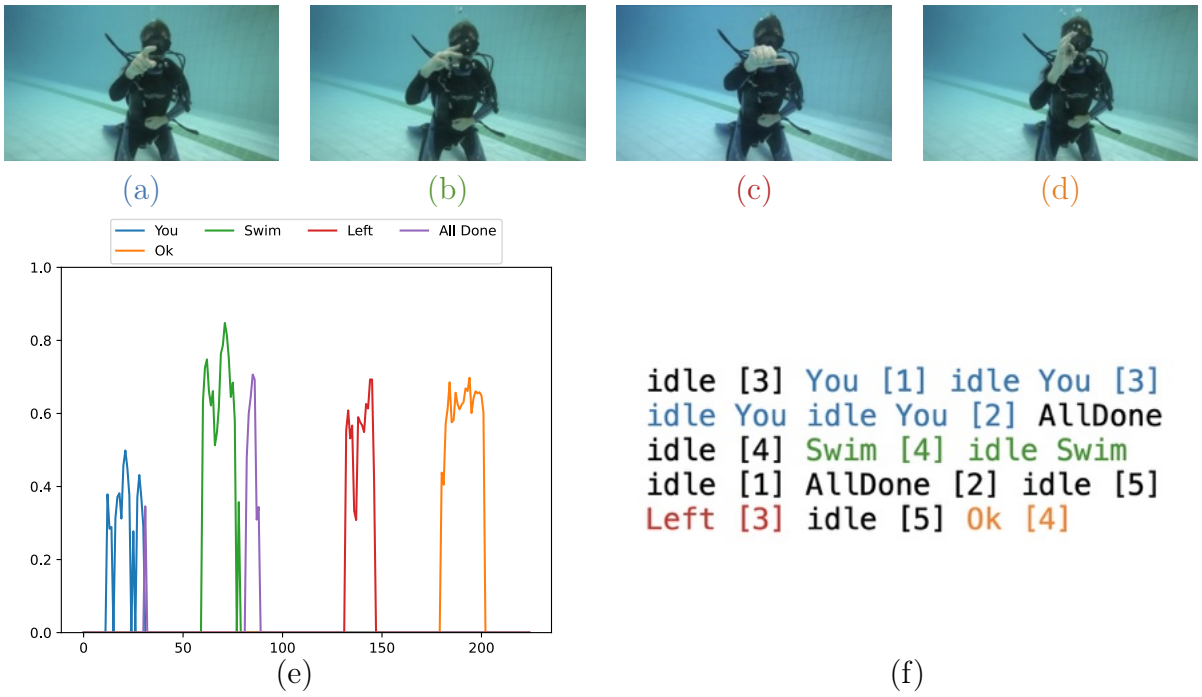


Figure 7.3: **Demo Gesture Encoding**
(a-d) shows keyframes in a sequence of the diver gesturing the phrase "You" "Swim" "Left" "Ok". (e) time series graph of the confidence values of all gestures detected in the sequence. (f) text encoding of the entire sequence fed into the SCUBALang LSTM resulting in the translated phrase – "You Swim Left Ok". This figure is color coded to illustrate when the gestures depicted in the keyframes appear in the expression.

Figure 7.1c shows the state of both the primary and secondary buffer after this process. In this case, an "Ok" gesture was appended to the secondary buffer so its contents are fed into the SCUBALang LSTM for translation. Figure 7.3 shows the result of this process for a demonstration sequence of the phrase "You Swim Left Ok". Key frames depicting the gestures within the seqeunce are shown in Figure 7.3(a)-(d). Figure 7.3(e) shows a time series graph of the per gesture confidence values outputted from SCUBANetV2. The run-length encoded gesture sequence fed into the LSTM for translation is shown in Figure 7.3(f).

Figure 7.3 summarizes the process of labelling individual frames using SCUBANetV2 and preparing the labels for processing using SCUBALang. Figure X(a)-(d) provides snapshots of

(a) (b)

Figure 7.4: **Perspective Errors**
(a) A diver presenting the "Me" gesture mistaken for the "Idle" gesture due to the occlusion of their pointer finger. (b) A diver presenting the "You" gesture mistaken for the "One" gesture due to the relative pose of the camera.

the gesture sequence as the diver signals You Swim Left Ok. Figure X(e) shows the recovered SCUBANetV2 labelling. Figure 7.3(f) shows the process of embedding this information in the buffer and run length encoding the input. Note that the figure is colour coded to highlight the relationship between frames, SCUBANetV2 labelling and the RLE version of the input.

7.4 Maximizing Communication Accuracy and Performance

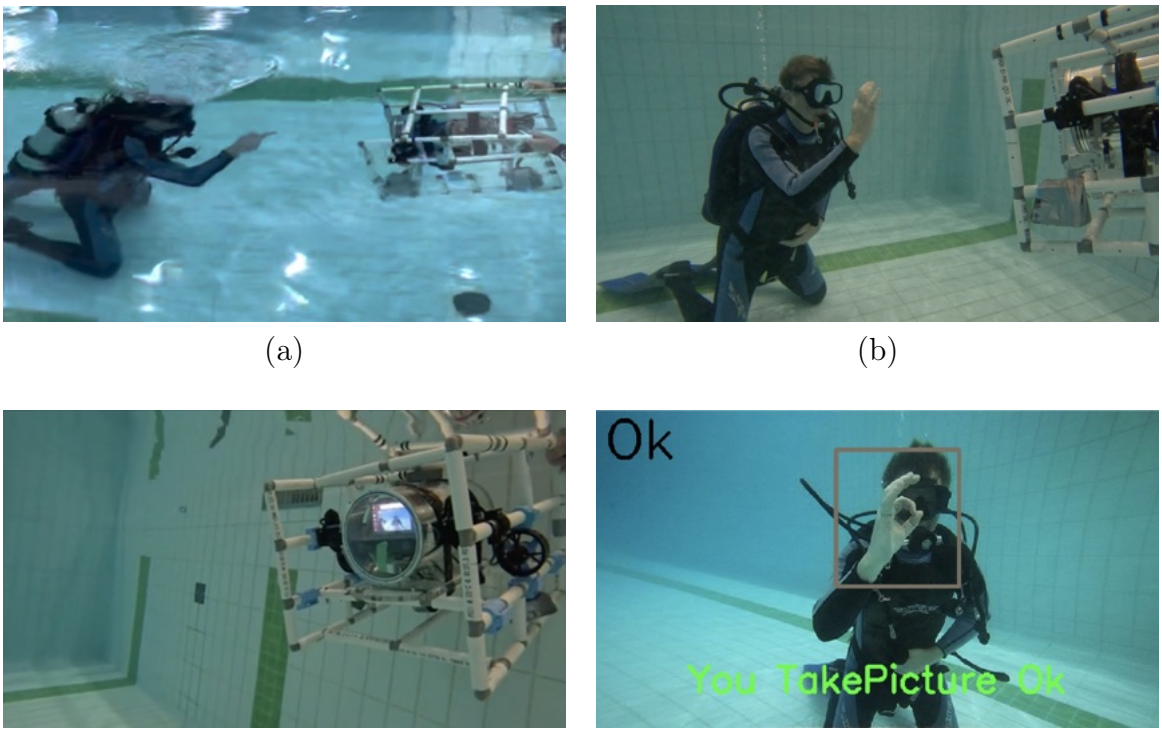
Communication failures are inevitable, whether they are caused by the robot misidentifying a gesture or a diver accidentally gesturing "Left" instead of "Right". It is important that the robot not act on erroneous commands as the consequences of such action can range from benign to catastrophic. In evaluating the performance of SCUBANetV2 and SCUBALang detection rates of 80% - 90% might seem sufficient. However, even one incorrect symbol in a gesture sequence can result in potentially catastrophic communication failure. The ease at which this sort of error can be introduced is illustrated in Figures 7.4 and 7.5. In

(a) (b)

Figure 7.5: **Eccentricity Errors**
(a) A diver presenting the "Three" gesture mistaken for the "Ok" gesture. (b) A diver presenting the "You" gesture mistaken for the "Idle" gesture.

Figure 7.4(a) the diver is attempting to present the gesture ME (which involves pointing at the chest) while in Figure 7.4(b) the diver is attempting to present the gesture You by pointing at the observer. Unfortunate choices of diver position can make it very difficult to recognize the specific gesture being presented. (Observe that missing the gesture You is particularly crucial given its use to book end a complete gesture sequence). Figure 7.5 illustrates the issue of similar gestures. In Figure 7.5(a) the diver is trying to present the gesture three, which because of the position of the diver's fingers can be easily confused with the symbol for ok. Figure 7.5(b) shows a similar confusion between an idle gesture and the gesture for Me. Dealing with these types of issues are critical for a fault-free communications strategy. To maximize communication performance, the recognition process is placed within an interaction loop to both improve individual gesture recognition as well to confirm entire gesture sequence commands.

The ability to provide feedback is an important part of establishing a robust communication protocol. Without this capability it is difficult and perhaps impossible for the sender to know whether messages transmitted by the sender are being properly received by the recipient.



(a) (b)

(c) (d)

Figure 7.6: **Milton and Diver**
(a) An overhead view of a diver interacting with Milton. (b) A diver gesturing to Milton. (c) A view of the Milton UUV and its front facing LCD panel. (d) A screen dump of Milton's display during interaction showing the camera view and three UI elements (current gesture [top left], detected phrase [center], gesture bounding box).

The Milton robotic platform is equipped with a 5 inch LCD display located just above the stereo camera that can be used to provide visual feedback to the diver. See Figure 7.6. The user interface shown in Figure 7.6(c) was designed to be readable at depth and limit onscreen clutter. Three types of visual feedback mechanisms are used to enhance the reliability of communication between the robot and diver as shown in Figure 7.7.

1. **Providing Continuous Feedback**: The display provides the diver ongoing information as to how the robot is perceiving and interpreting their actions. This includes a live camera feed to the diver of the robot's view that is annotated with the currently detected gesture class in the top left corner and a bounding box around the gesture

with the highest confidence score. Additionally when the number of gestures in the primary buffer surpasses a threshold β the gesture class is displayed on the lower portion of the screen in magenta. This process is illustrated in Figure 7.7 which shows the location of the feedback information and the transitions encountered by the diver as they communicate with the robot.

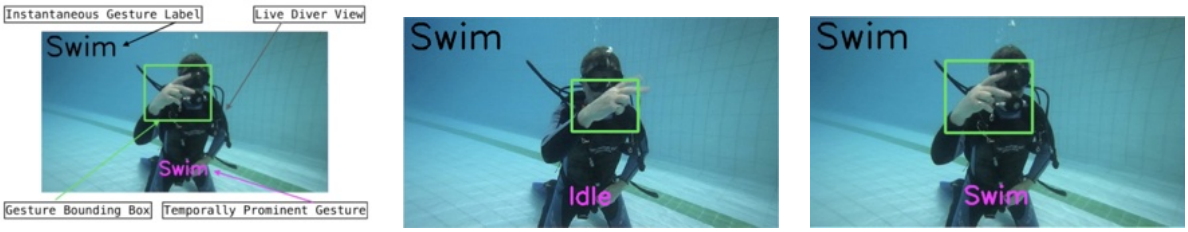
2. **Per Gesture-Sequence Feedback**: Once a full sequence has been recognized, or the sequence has been determined to be invalid, the robot displays the detected phrase in the center of the screen. If the phrase is book-ended with the "You" and "Ok" gestures it is deemed valid and displayed in green, invalid phrases are displayed in red. Incomplete sequences are left until they eventually fall off the buffer.

3. **Communication Confirmation**: Once a valid sequence has been processed, the robot seeks confirmation prior to executing the command. This interaction enables the diver to indicate to the robot that last phrase was detected correctly by communicating the phrase "You" "Ok".

The technical details of each of these interactions are described in detail below.

7.4.1 Providing Continuous Feedback

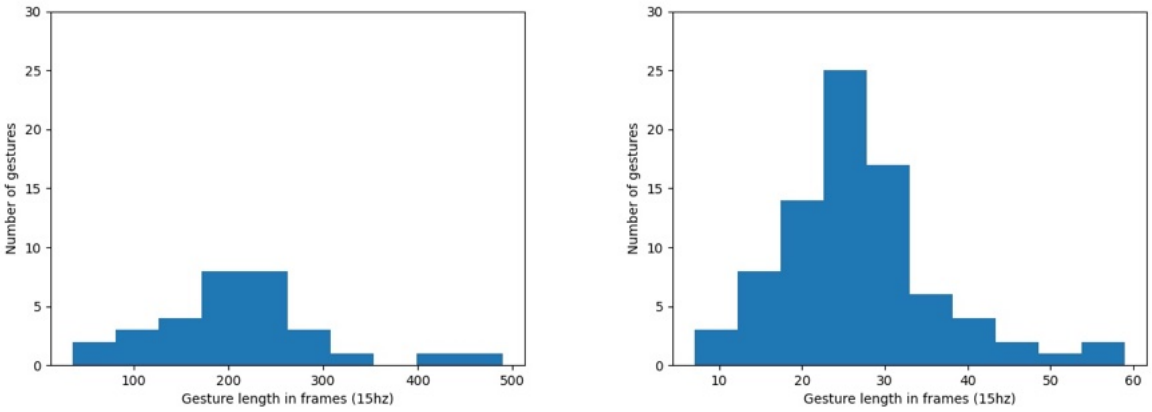
As described earlier, the SCUBANetV2 YOLO model has difficulty detecting some gestures due to various factors. For example, the pose of the camera with respect to the gesturing hand can cause some important features associated with the gesture to become occluded. This occurs with the "You" and "Me" gestures as shown in Figure 7.4. Personal eccentricities regarding the presentation of gestures can also have a deleterious effect on gesture detection as well. For example, some divers present the gesture "Three" using their middle, index and



(a) (b) (c)

Figure 7.7: **Providing Continuous Feedback**
Shows the continuous feedback a diver receives regarding the detection of gestures being presented to the robot. This is displayed on Milton's forward facing monitor. (a) annotated image showing the placement of various UI elements on the screen presented to the diver (b) Show the current gesture "Swim" detected by the SCUBANetV2 model in the top left corner and the last gesture to appear in significant quantity inside the primary buffer is shown in magenta in the center of the lower portion of the screen. (c) As the number of gestures in the primary buffer passes a set threshold the gesture – in this case "Swim" – is displayed to the diver in magenta in the center of the lower portion of the screen.

pinky fingers which can be mistaken for the "Ok" gesture, others use two or more fingers to present the "You" gesture as show in Figure 7.5. Repeated communication failures can cause frustration, especially when there is no indication as to why the failure could be happening. Displaying the live camera feed to the diver enables the diver to properly position themselves within the frame at an appropriate distance and present gestures more clearly to the robot. But other feedback is also provided. Divers need to be informed about incorrect gesture detection so that they can perform corrective actions on the fly, such as repositioning a pointer finger for more clarity or re-configuring their presentation of a gesture to better conform to the standard definition. A simple way of providing this additional feedback is to present the diver with annotated output of the SCUBANetV2 model as shown in Figure 5.3. However, this would not be ideal as detection beyond the dominate hand would clutter the display and attaching the gesture label to the moving bounding box would hinder the diver's ability to read it. Instead an approach better suited for the diver operating at depth is used. The live camera feed is only annotated with a single bounding box around the dominant



(a) Gesture duration without Feedback (b) Gesture duration using Feedback

Figure 7.8: **Gesture Duration Comparison**
(a) Shows the distribution of the duration of gestures in all successfully detected phrases without using feedback from the slow dataset. (b) Shows the distribution of the duration of gestures in all successfully detected phrases using feedback. The units of the horizontal axis is expressed in number of frames (1 second = 15 frames). Note the different horizontal scales in the two graphs.

hand and the gesture detected in the current frame is displayed in the top left hand corner in a large font (see Figure 7.7(a)). This continuous feedback mechanism is intended to condition divers to communicate slower and with increased clarity as they integrate feedback into their gesturing. Figure 7.8 shows the effect this feedback mechanism has on the average length of gestures in a successfully detected phrase in comparison with experiments in the previous chapter where divers were asked to hold the gesture for five seconds. Sequences are presented approximately ten times faster when feedback is provided.

7.4.2 Providing per Gesture-Sequence Feedback

The SCUBALang LSTM model can have difficulty deciphering the intended expression when divers present gestures for only a short period of time or when they quickly transition from

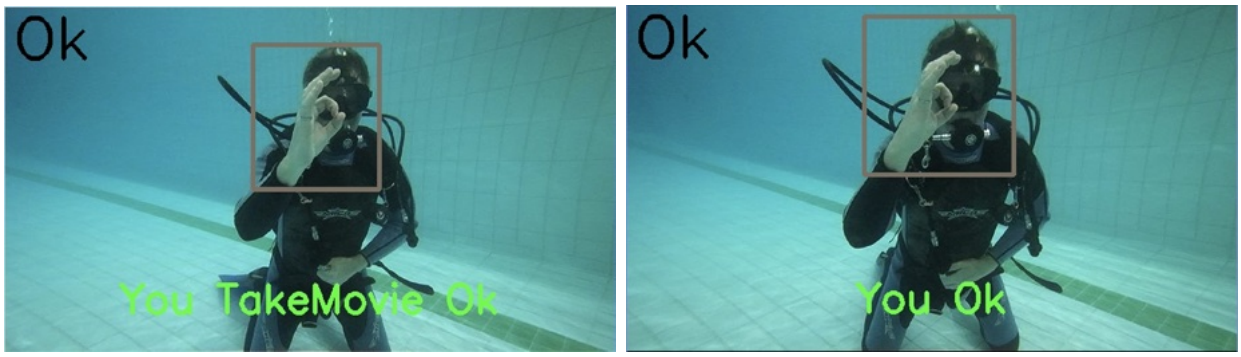
(a) Invalid (b) Valid

Figure 7.9: **Providing per Gesture-Sequence Feedback**
(a) the robot has detected an invalid phrase, displayed in red to the diver (b) the robot has detected a valid phrase (i.e., book-ended by "You" and "Ok" gestures), displayed in green to the diver.

one gesture to another without a sufficient pause or idle period in between gestures. While the continuous feedback mechanism described above is intended to reduce the occurrences of these events some could still persist and other circumstances may introduce errors. In order to provide a diver confirmation that a gesture sequence was recognized properly, after each completed sequence this feedback mechanism provides the diver with the completed phrase as detected by the SCUBALang LSTM model (see Figure 7.9). Valid phrases start with the gesture "You" and end with "Ok" and are displayed in green to the diver, invalid phrases are displayed in red. Incomplete phrases stay on the secondary buffer until they are eventually removed to make room for new detections.

7.4.3 Communication Confirmation

That the UUV has received a valid command does not necessarily mean that the command is what the diver intended. The diver needs to be able to review a communicated command sequence and to indicate whether or not the sequence is what was intended. It may seem

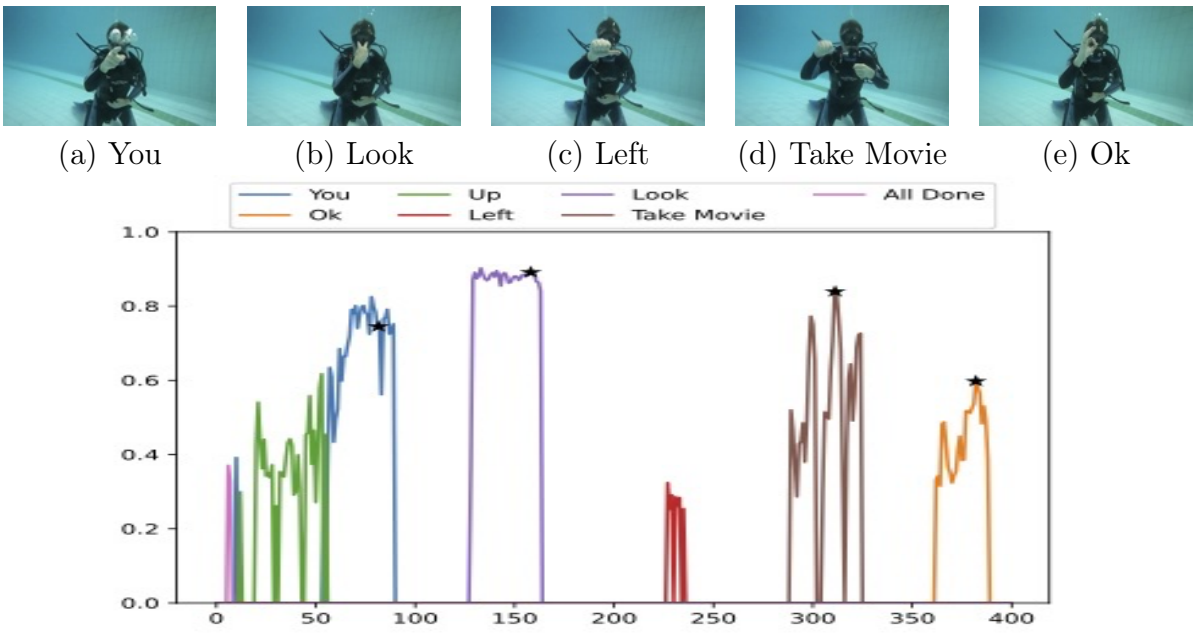


(a) Request (b) Confirmation

Figure 7.10: **Communication Confirmation**
(a) The robot signals to the diver it has detected the phrase "You" "TakeMovie" "Ok" (b) After which the diver responds with the phrase "You" "Ok" the robot begins executing the last detected phrase, if valid.

natural to use idiomatic gestures such as *thumbs up* and *thumbs down* to indicate success or failure as they are already included in the set of identifiable gestures. However in this context those gestures represent "Up" and "Down", while not prohibitory in its own right this could be a source of confusion. Nor would it be possible to encode these short phrase in SCUBALang as these standalone gestures do not constitute valid SCUBALang phrases. Such sequences would require a different mechanism for their detection after a phrase has been detected by the model. Converting these gestures to valid phrases by prepending the "You" gesture and appending the "Ok" gesture changes their meaning as "You" "Up" "Ok" can be used to signal the end of a dive.

The shortest valid phrase "You" "Ok" provides the versatility needed in this circumstance; it can be performed quickly, it has a high detection rate in testing, and it has limited use in human-to-robot communication. This phrase can be used to inform the robot that it has successfully identified the intended phrase/command and to proceed with executing the action (see Figure 7.10 for example). There is no for a separate phrase to indicate failure, if



(a) You (b) Look (c) Left (d) Take Movie (e) Ok

(f) time series of SCUBANetV2 output

Figure 7.11: **Demo Gesture Sequence**
(a-e) shows keyframes in a sequence of the diver gesturing the phrase "You" "Look" "Left" "Take Movie" "Ok". (f) time series graph of the confidence values of all gestures detected in the sequence, a star indicated the moment when a diver was signaled that the number of symbols in the primary buffer surpassed the β threshold. Note, in this sequence the diver was not signaled for the "Left" gesture.

a communication failure has occurred the diver can repeat the phrase.

7.5 Demonstration

The effectiveness of the integration of these strategies was demonstrated in a pool trial using a SCUBA diver who appears in the training dataset but had not interacted with the full system previously. After a short introductory period (5 trials) the robot was successfully detecting complete phrases at a rate of over 65% on the first attempt (35% failure rate) over 21 trials (see Table 7.1). Using this as a predictor of future performance on a given trial run, a second and third attempt would yield failure rates of 12% and 4% respectively. The

Expression	Status	Details
You Look Right Ok	Y	
You Look Left Ok	N	Left gesture was not properly detected, mistaken for Stop/Idle
You Swim Left Ok	N	Left gesture was not properly detected, mistaken for Idle
You Look Left Ok	Y	
You Swim Right Ok	Y	
You Swim Left Ok	N	Left gesture was not properly detected, mistaken for Stop/Idle
You Swim Left Ok	Y	
You Swim Right Ok	Y	
You Look Right TakePic Ok	N	Instability in the detection of TakePic gesture caused a double detection
You TakePic Ok	N	Instability in the detection of TakePic gesture caused a double detection
You Look Right TakeMovie Ok	Y	
You TakeMovie Ok	Y	
You Swim Left Ok	Y	
You Swim Down Ok	Y	
You Swim Up Ok	Y	
You TakePic Ok	N	Instability in the detection of TakePic gesture caused a double detection
You TakePic Ok	Y	
You Look Left TakePic Ok	Y	
You TakePic Ok	N	Instability in the detection of TakePic gesture caused a double detection
You Swim Right One Ok	Y	
You Swim Left One Ok	Y	

Table 7.1: **Demonstration Sequences**
A list of gesture phrases used in the demonstration accompanied by a 'Y' indicating successful detection or 'N' indicating failure and an explanation for the cause of the failed detection.

number of Bernoulli trials until the first success $1/p(p = 0.65) = 1.5$. That is, on average the diver will be able to communicate with the robot and acknowledge success in under two attempts. A number that would be expected to improve with experience. The majority of incorrect detection only contained a single gesture error, a significant improvement over the non-interactive test sequences described in Chapter 6. Where the average failure rate p=65%, indicating on average that 2.2 attempts would be recorded, and without the benefit of feedback to understand that the communication had, or had not, been successful. When

provided with in-gesture sequence feedback, the diver often made corrective actions while presenting gestures in response to information from the continuous feedback mechanism.

Figure 7.12 shows an example of this. The diver presents the gesture "Left" which is mistaken for the "Stop" gesture. Seeing this, the diver slightly alters their presentation of the gesture so that "Left" is successfully detected. The diver maintains this pose until the robot indicates the gesture was detected for an adequate period of time.

Figures 7.11- illustrate the interactive process in operation. Figure 7.11 shows a breakdown of a successfully detected phrase – "You Look Left Take Movie Ok" – from the demonstration. Figure 7.11a-e shows keyframes from the sequence of the gestures that makeup phrase. Figure7.11f shows a time series graph of the per gesture confidence values outputted by the SCUBANetV2 model. This graph is annotated with asterisks that shows when the current gesture was presented to the diver as temporarly stable. In this case the gestures "Up" "You" "Look" "Take Movie" "You" were indicated to be temporarly stable. Notably the gesture "Left" was not indicated to the diver as being temporally stable but was correctly detected. The short presentation of this gesture can be attributed to the diver already having performed a corrective action for the gesture during the previous phrase.

Feedback from the UI to the diver allows the diver to take corrective action to ensure that the robot is recognizing the intended gesture.

7.6 Summary

Gesture based spatiotemporal action recognition models typically require a large memory/GPU/CPU footprint. By decoupling the problem into a spatial detection problem and a temporal translation problem, both problems can be addressed using neural network architectures already established in the literature. These models can be optimized using

(a) Incorrect Detection (b) Corrective Action (c) Correct Detection

Figure 7.12: **Corrective Action using Feedback**
(a) shows the diver continuing to present a "Left" gesture mistaken for a "Stop" gesture. (b) shows the diver augmenting their presentation of the "Left" gesture to accommodate the SCUBANetV2 model. (c) shows the diver continuing to present the "Left" gesture until enough detections have been added to the primary buffer that the diver can proceed to the next gesture in the intended sequence.

NVIDIA's TensorRT tools to run within computational limits of a real UUV

In any gesture-based language, communication errors will always be present. In order to ensure accuracy it is important to assist the human to communicate correctly. This system includes four features for this purpose. (a) It provides the diver live feedback to aid them in being properly aligned with the robot for effective communication. (b) It provides the diver live feedback of the specific gesture that the robot is detecting, and the temporal stability of that gesture. (c) It provides full-gesture feedback that ensures that all communication falls within the formal framework of a command to the robot. This also aids in the prevention of accidental communications or misunderstandings. Finally (d), it uses an acknowledgment process that ensures that the diver must explicitly acknowledge the communication before the robot executes the command given. During the system demonstration a diver with limited training learned to communicate with the robot so that 65% of all communication was successful on the first attempt.

This work has demonstrated how semi-supervised learning along with Sim2Real techniques can be used to create a SCUBA gesture recognition system from partially labeled data and a small amount of hand labelled data. This recognition process is then incorporated into

a human-robot interaction system to improve detection rates in comparison with test data obtained in the absence of interactions with the robot.

Chapter 8

Summary and future work

8.1 Summary

Underwater human-robot interaction is a difficult problem made more complex by a harsh environment. Particulate matter gets suspended in the water column reducing visibility, light is attenuated to a higher degree in air causing colors to fade with distance and depth. This attenuation is stronger for other frequencies in the electromagnetic spectrum such as radio and infrared rendering common wireless communication technologies ineffective. Furthermore, divers require special equipment to breath and maintain proper buoyancy which increases their cognitive load, so robots need simple interaction schemes.

Traditional approaches to this problem use tethered controllers and visual marker libraries. Tethered controllers with a display and a few input switches to interface with the robot. These systems are more bulky and less intuitive than their terrestrial counterparts due to their need to be water tight. Visual marker libraries use high contrast fiducial markers that map to specific commands which can be presented to the robot to form command sequences for execution. These libraries must trade-off ease of use for the diversity of commands. Modern

approaches to underwater human robot interaction have begun to employ gestures made possible by small form-factor high efficiency computing platform such as the Nvidia Jetson and advances in neural network-based computer vision algorithms. Current underwater gesture systems rely on augmenting the diver with visual markers to improve detection of gestures which reduces the generality of the approach and/or composing commands from a combination of a small set of distinguishable gestures requiring divers to learn this new language.

Gesture recognition systems need to solve a number of problems to operate effectively. The first task is designing the gesture language that is going to be used for communication. A large gesture set can make the language more expressive but has implication for the complexity of the system and the ability of a new user to learn to communicate effectively. Highly distinguishable gestures can be chosen to the benefit of the recognition system or representative gestures can be chosen to make them easier to remember for the user. The only true requirement here is that gestures must easy to perform. Gesture recognition systems generally take one of two routes in this regard; borrow a subset of gestures and their meaning from a larger established gesture language such as ASL, or select a small set of distinguishable gestures and assign meaning to them in accordance with the needs of the system. The next task follows from the first, detecting individual gestures, the complexity of this task is determined by the size of the gesture set and the defining features of each gesture. Traditional computer vision techniques such as contour analysis or template matching can be employed effectively on small gesture sets with simple defining characteristics. These techniques fall short when dealing with larger gesture sets with more nuanced differences between gestures which benefit from data-driven approaches. As gesture recognition can be formulated as an object detection or action recognition problem, neural networks designed for these problems

can be retooled for this purpose given enough training data. As these types of neural networks require a large amount of training data the problem becomes acquiring and labeling this data, a not so simple task especially in the underwater domain. Once individual gestures can be accurately detected this temporal sequence needs to be processed to determined the sequence of gestures in the phrase the user is communicating to the robot. The process must be robust to noise caused by failures in gesture detection and minor mistakes in gesture presentation on the part of the diver. Furthermore if the gesture language gives significance to repeated gestures this process must determine the difference between a repeated gesture and a single gesture held for a longer duration in the presence of noise. To deploy a gesture recognition system in a human-robot interaction setting there needs to be way to determine whether gestures have been properly interpreted by the system. In some cases it may be desirable to immediately respond to gestures through action. However, communication failures and the potential consequences of incorrect actions limit the usefulness of highly responsive systems. As failure detection is an essential component of reliable communication and as any system is limited in its ability to detect failure, user should be provided the opportunity to clarify, repeat or correct any miscommunications.

This work presents a gesture system for underwater human robot communication that leverages a gesture language divers already know and use to communicate with each other. Utilizing the standard SCUBA gesture language is of substantial benefit to divers and to the usefulness of the system as a whole. Individual gesture detection is accomplished employed a data-driven approach using the YOLOv8 object detection neural network model. This model was trained on data acquired in a structured manner and labeled using weak-supervision methods to reduce the amount of manual labelling required. This weak-supervision system employs the crowd-sourced fully supervised object detection model SCUBANetV1+ described

in Chapter 4 to identify salient diver body parts and combines them with image classification labels using a set of heuristics to create a much larger dataset of diver hand gestures.

Chapter 5 shows how SCUBANetV1+ can be leveraged in a semi-supervised learning algorithm to label a much larger dataset of SCUBA hand gestures – for the purposes of training a gesture detection model (SCUBANetV2) – using the start and end times for gestures repeated in short video clip. This dataset is used to train a YOLOv8 based object detection model referred to as SCUBANetV2.

Chapter 6 describes a method to reuse these video clips to create a vast dataset of synthetic gesture sequences/phrases in order to help bridge the Sim2Real gap and train a long short-term memory recurrent neural network capable of translating the output of SCUBANetV2 into discernible gesture phrases. Exceptional performance on simulated data sequences did not translate directly to natural gesture sequences provided by divers. An examination of errors in SCUBANetV2 labelling suggested that mechanisms should be applied that encouraged divers to present gestures clearly and for of sufficient duration to be captured.

Beyond enhancing the process of the application of SCUBANetV2 to gesture sequences it is also of interest to enhance SCUBANetV2's performance more generally. Traditional approaches to improving the detection rates of these models such as collecting more data or curating the data to remove outliers, correct labels and improve bounding would require a significant amount of work, but would offer significant improvements. In the preparation of the dataset for training SCUBANetV2 a large number (140,000) of "Idle" frames were randomly culled to prevent overfitting. While a number of these frames are very highly correlated with other gestures due the structure of the data collection and annotation process, a significant number are representative of an "Idle" gesture. Dilating the temporal segmentation of each gesture could be enough to partition good "Idle" frames from those that correlate highly with

a gesture. Another possibility is to remove "Idle" and "Off-hand" from the set of class labels and instead use these as negative examples.

Expanding the dataset used for training the body part detector SCUBANetV1 to improve the detection rate or the placement of bounding boxes of the hand class would improve the quality of data in the resulting SCUBANetV2 dataset. This could be accomplished using semi-supervised learning techniques (see [115]) or simply manually labeling some of the videos collected for SCUBANetV2. In certain applications, it would also be of interest to tune SCUBANetV2 to the personal preferences of a given diver in terms of their gesture presentations. Sim2Real techniques were used to create synthetic sequences of divers performing gesture phrases. The output of SCUBANetV2 over the course of these sequences was used to train an LSTM neural network to translate temporal gesture detections into gesture phrases, this LSTM is referred to as SCUBALang. Together SCUBANetV2 and SCUBALang form a neural network pipeline that can be used to recognize diver hand gestures in video data. It was demonstrated that this approach works to recognize well structured gesture sequences but that extra effort must be used to ensure that divers work to provide clear gestures, and that a feedback mechanism can provide this information to reduce the number of gesture attempts to an average of approximately 1.5 attempts. The SCUBALang LSTM sometimes produces translated phrases that do not conform to the formal structure of a command to the robot (i.e., start with "You" and end with "Ok"). This is an indication that LSTM has not fully learned the structure of the language. The set of training phrases, could be expanded by defining a formal grammar for the language to enumerate a list of all valid phrases of length N. The Sim2Real approach described in Chapter 6 could be used to generate a large number of test sequences for each valid phrase. A similar approach could be used to generate negative examples for the LSTM aswell.

Different methods to encode gestures for input into the LSTM could also yield promising results. Instead of compressing the stream other signal processing techniques could be used, such as noise filtering and frequency analysis. More information from the object detection phase could also be included such as confidence values or spatial relationships of the gestures to the head or other hand. Futhermore the use of the LSTM architecture is not a requirement, any neural network architecture capable of processing sequences could be used to varying degrees of success. Other Recurrent neural networks like the GRU (gated recurrent unit) or even attention-base transformer models would make for a good substitute for the LSTM in this case.

During the course of this research, several standalone components have been developed:

- designed and constructed the Milton unmanned underwater robotic platform.
- manually labeled SCUBA diver datasets for diver body parts.
- Developed the SCUBANetV1+ dataset and object detection model to recognized salient diver body part (hands, head and body) in underwater imagery.
- Developed the SCUBANetV2 weakly labelled dataset and object detection model to recognized SCUBA hand signals in underwater imagery.
- Developed the SCUBALang synthetic dataset and LSTM neural network to translate the temporal output of SCUBANetV2 into gesture phrases performed by divers in a video sequence.
- Developed a weak supervision approach that has wide application in terms of enlarging the set of labels through controlled data collection.

- Developed a sim2real approach that leveraged the very large labelled dataset to build realistic — in terms of SCUBANetV2 labels — gesture sequences dataset that can be generated to train the LSTM network SCUBALang for gesture recognition.
- Developed a feedback-based system that provides robust reliable diver->robot communication at depth.
- Completed a test of the entire interaction system with a SCUBA diver in a real-world environment.

8.2 Future Work

While this research has demonstrated the effectiveness of the gesture recognition framework developed here, more extensive user studies with both commercial and recreational divers would be very beneficial. These studies could provide a more comprehensive understanding of the system's usability, reliability, and overall performance in diverse real-world conditions.

For instance, involving a larger group of commercial divers could offer insights into the framework's practicality and efficiency in professional settings where precision and quick communication are crucial. Their feedback could help refine the system to better meet the demanding requirements of underwater construction, inspection, and repair tasks.

On the other hand, testing with recreational divers could reveal how user-friendly and accessible the gesture recognition system is for casual users who may have varying levels of experience and expertise. Such studies could help identify potential areas for improvement to enhance the overall diving experience, making underwater communication more intuitive and enjoyable.

Another avenue to explore is the application of this approach to gesture languages in

the terrestrial domain. Gesture languages, such as sign languages used by the deaf and hard-of-hearing communities, as well as various application specific gesture languages used in specific profession, hold significant importance for effective human interaction. The SCUBANetV1+ model, which is currently employed in underwater environments for specific detection tasks, could be replaced with one of a number of sophisticated hand or body part detection mechanisms. These mechanisms could include advanced models such as OpenPose, MediaPipe, or other state-of-the-art machine learning frameworks specifically designed for detecting salient body parts. This would involve utilizing the weak-supervision process described here, wherein large amounts of unlabeled data can be efficiently annotated with minimal human intervention.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *In Proceedings OSDI'16 12th USENIX Conference on Operating Systems Design and Implementation, Savannah, GA*, 2016, pp. 265–283.

[2] I. Asimov, *I, Robot*. Harper Voyager, 2018, ISBN: 978-0-00-736935-5. [Online]. Available: `https://harpervoyagerbooks.co.uk/products/i-robot-isaac-asimov-9780007369355/`.

[3] J. Baglio, *My experiences from learning ASL*, [Accessed] August, 2019, Sep. 2017. [Online]. Available: `https://www.theodysseyonline.com/my-experiences-from-learning-asl`.

[4] K. Bansal, G. Reddy, and D. Bharadia, "Shenron - scalable, high fidelity and efficient radar simulation," *IEEE Robotics and Automation Letters*, vol. 9, pp. 1644–1651, 2024.

[5] R. Barandela and E. Gasca, "Decontamination of training samples for supervised pattern recognition methods," in *Joint IAPR International Workshop on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, Alicante, Spain, 2000, pp. 621–630.

[6] L. Behnke, *SCUBA diving hand signals: underwater communication pocket companion for recreational scuba divers*. Barns and Noble, 2015, ISBN: 978-1-5024-8848-0. [Online].

Available: https://www.barnesandnoble.com/w/scuba-diving-hand-signals-lars-behnke/1121720575.

[7] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, 1994.

[8] *Biological neuron model*, Nov. 2019. [Online]. Available: https://en.wikipedia.org/wiki/Biological_neuron_model.

[9] A. Birk, "A survey of underwater human-robot interaction (u-hri)," *Current Robotics Reports*, vol. 3, no. 4, pp. 199–211, 2022.

[10] BlueRobotics. "Bluerov (retired) - bluerobotics." Accessed 30-April-2017. (2015), [Online]. Available: https://www.bluerobotics.com/store/retired/bluerov-r1/.

[11] *Bluerov2 - affordable and capable underwater drone*, [Accessed] August, 2019. [Online]. Available: https://bluerobotics.com/store/rov/bluerov2/bluerov2/.

[12] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998, pp. 92–100.

[13] J. Branson, D. Miller, I. G. Marsaja, and I. W. Negara, "Everyone here speaks sign language, too: A deaf village in Bali, Indonesia," *Multicultural Aspects of Sociolinguistics in Deaf Communities*, vol. 2, pp. 39–57, 1996.

[14] B. Brumfield, *Car assembly line robot kills worker in Germany*, Jul. 2015. [Online]. Available: https://www.cnn.com/2015/07/02/europe/germany-volkswagen-robot-kills-worker/index.html.

[15] M. Brundage, "Taking superintelligence seriously: Superintelligence: Paths, dangers, strategies by nick bostrom (oxford university press, 2014)," *Futures*, vol. 72, pp. 32–35, 2015.

[16] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017, pp. 1302–1310.

[17] S. Chamorro, J. Collier, and F. Grondin, "Neural network based lidar gesture recognition for realtime robot teleoperation," in *2021 International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, New York, USA, 2021, pp. 98–103.

[18] D. Chiarella, M. Bibuli, G. Bruzzone, *et al.*, "Gesture-based language for diver-robot underwater interaction," in *OCEANS 2015 - Genova*, 2015, pp. 1–9.

[19] D. Chiarella, M. Bibuli, G. Bruzzone, *et al.*, "A novel gesture-based language for underwater human–robot interaction," *Journal of Marine Science and Engineering*, vol. 6, p. 91, 2018.

[20] R. Codd-Downey and M. Jenkin, "Finding divers with scubanet," in *IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, 2019, pp. 5746–5751.

[21] R. Codd-Downey and M. Jenkin, "Recognizing diver hand gestures for human to robot communication underwater," in *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, Busan, South Korea, 2023.

[22] R. Codd-Downey, M. Jenkin, and K. Allison, "Milton: An open hardware underwater autonomous vehicle," in *IEEE International Conference on Information and Automation (ICIA)*, Macau SAR, China, 2017, pp. 30–34.

[23] R. Codd-Downey and W. Stuerzlinger, "Leaplook: A free-hand gestural travel technique using the leap motion finger tracker.," in *ACM Symposium on Spatial User Interaction (SUI)*, Honolulu, USA, 2014, p. 153.

[24] *Common hand signals for recreational scuba diving*, `http://www.neadc.org/CommonHandSignalsforScubaDiving.pdf`, Accessed: 2018-09-2.

[25] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 1, San Diego, USA, 2005, pp. 886–893.

[26] C. De Souza, A. Gaidon, Y. Cabon, and A. Lopez, "Procedural generation of videos to train deep action recognition networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, 2017, pp. 2594–2604.

[27] T. Defanti and D. Sandin, "Final report to the national endowment of the arts," *US NEA R60-34-163, University of Illinois at Chicago Circle, Chicago, Illinois*, 1977.

[28] A. Diba, M. Fayyaz, V. Sharma, *et al.*, "Temporal 3d convnets using temporal transition layer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Salt Lake City, USA, 2018, pp. 1117–1121.

[29] *Diver communications*, [Accessed] April, 2021, Mar. 2021. [Online]. Available: `https://en.wikipedia.org/wiki/Diver_communications#RSTC_hand_signals`.

[30] J. Donahue, L. Anne Hendricks, S. Guadarrama, *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, 2015, pp. 2625–2634.

[31] G. Dudek, J. Sattar, and A. Xu, "A visual language for robot control and programming: A human-interface study," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, Apr. 2007, pp. 2507–2513. [Online]. Available: http://cim.mcgill.ca/mrl/pubs/anqixu/icra2007_robochat.pdf.

[32] C. E. B. an E. Carla E and M. A. Friedl, "Identifying mislabeled training data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[33] E. Eakin, *Before the word, perhaps the wink?; some language experts think humans spoke first with gestures*, May 2002. [Online]. Available: https://www.nytimes.com/2002/05/18/books/before-word-perhaps-wink-some-language-experts-think-humans-spoke-first-with.html.

[34] M. Everingham, A. Zisserman, C. K. I. Williams, *et al.*, "The 2005 Pascal visual object classes challenge," in *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché-Buc, Eds., Springer Berlin Heidelberg, 2006, pp. 117–176.

[35] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, pp. 1933–1941.

[36] M. Fiala, "ARTag, a Fiducial Marker System using Digital Techniques," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2005, pp. 590–596.

[37] T. Foote and R. Rusu. "Nodelet - ros wiki." Accessed 1-May-2017. (2014), [Online]. Available: http://wiki.ros.org/nodelet.

[38] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "ActionVLAD: Learning spatio-temporal aggregation for action classification," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017, pp. 971–980.

[39] R. Girshick, "Fast R-CNN," pp. 1440–1448, 2015. [Online]. Available: `http://arxiv.org/abs/1504.08083`.

[40] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," pp. 580–587, 2014. [Online]. Available: `https://arxiv.org/abs/1311.2524`.

[41] A. Gomez Chavez, C. Mueller, T. Doernbach, D. Chiarella, and A. Birk, "Robust gesture-based communication for underwater human-robot interaction in the context of search and rescue diver missions," Oct. 2018.

[42] H. Hassan, Z. Ren, C. Zhou, *et al.*, "Supervised and weakly supervised deep learning models for COVID-19 CT diagnosis: A systematic review," *Computer Methods and Programs in Biomedicine*, p. 106 731, 2022.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, 2016. [Online]. Available: `http://arxiv.org/abs/1512.03385`.

[44] D. Horváth, G. Erdős, Z. Istenes, T. Horváth, and S. Földi, "Object detection using sim2real domain randomization for robotic applications," *IEEE Transactions on Robotics*, vol. 39, pp. 1225–1243, 2023.

[45] H. Hwang, C. Jang, G. Park, J. Cho, and I.-J. Kim, "Eldersim: A synthetic data generation platform for human action recognition in eldercare applications," *IEEE Access*, vol. 11, pp. 9279–2023, 2023.

[46] *Imagenet*, [Accessed] August, 2019, Jan. 2016. [Online]. Available: https://www.optitrack.com.

[47] M. J. Islam, M. Ho, and J. Sattar, "Dynamic reconfiguration of mission parameters in underwater human-robot collaboration," in *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, 2018, pp. 6212–6219.

[48] Y. Jia, E. Shelhamer, J. Donahue, *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ACM, New York, USA, 2014, pp. 675–678.

[49] G. Jocher, A. Chaurasia, and J. Qiu, *YOLO by Ultralytics*, version 8.0.0, Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[50] A. Kadian, J. Truong, A. Gokaslan, *et al.*, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, pp. 6670–6677, 2020.

[51] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1725–1732.

[52] A. Khazri, *Faster rcnn object detection*, Apr. 2019. [Online]. Available: https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4.

[53] *Kinect*, [Accessed] August, 2019, Jul. 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Kinect.

[54] A. Kirillov, E. Mintun, N. Ravi, *et al.*, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].

[55] K. Koreitem, J. Li, I. Karp, T. Manderson, and G. Dudek, "Underwater communication using full-body gestures and optimal variable-length prefix codes," in *IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, 2019, pp. 8018–8025.

[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in ACM, 2017, vol. 60, pp. 84–90.

[57] A. Kuznetsova, H. Rom, N. Alldrin, *et al.*, "The Open Images Dataset V4," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[58] I. Kvasić, D. O. Antillon, Đ. Nađ, C. Walker, I. Anderson, and N. Mišković, "Diver-robot communication dataset for underwater hand gesture recognition," *Computer Networks*, vol. 245, p. 110 392, 2024.

[59] *Leap motion*, [Accessed] August, 2019, Jul. 2019. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Leap_Motion`.

[60] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, 1988, pp. 21–28.

[61] J.-W. Lee and K.-H. Yu, "Wearable drone controller: Machine learning-based hand gesture recognition and vibrotactile feedback," *Sensors*, vol. 23, 2023.

[62] M. Li and Z.-H. Zhou, "Setred: Self-training with editing," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, Hanoi, Vietnam, 2005, pp. 611–621.

[63] M. Li and Z.-H. Zhou, "Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 6, pp. 1088–1098, 2007.

[64] S. K. Liddell, *Grammar, Gesture, and Meaning in American Sign Language*. Washington DC, USA: Cambridge University Press, 2003.

[65] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft COCO: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, Zurich, Switzerland, 2014, pp. 740–755.

[66] *List of sign languages by number of native signers*, Nov. 2019. [Online]. Available: `https://en.wikipedia.org/wiki/List_of_sign_languages_by_number_of_native_signers`.

[67] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[68] Z. Lu and E. Elhamifar, "Weakly-supervised action segmentation and alignment via transcript-aware union-of-subspaces learning," in *IEEE International Conference on Computer Vision (ICCV)*, Montreal, Canada, 2021, pp. 8085–8095.

[69] D. McNeill, *Hand and mind: What Gestures Reveal About Thought*. University of Chicago Press, 1992. [Online]. Available: `https://www.press.uchicago.edu/ucp/books/book/chicago/H/bo3641188.html`.

[70] W. Meeussen. "Ros_control - ros wiki." Accessed 1-May-2017. (2014), [Online]. Available: `http://wiki.ros.org/ros%5C_control`.

[71] Q. Miao, Y. Lv, M. Huang, X. Wang, and F.-Y. Wang, "Parallel learning: Overview and perspective for computational learning across syn2real and sim2real," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, pp. 603–631, 2023.

[72] N. Mišković, A. Pascoal, M. Bibuli, *et al.*, "CADDY project, year 3: The final validation trials," in *OCEANS 2017 - Aberdeen*, Jun. 2017, pp. 1–5.

[73] M. Monajjemi, J. Bruce, S. A. Sadat, J. Wawerla, and R. Vaughan, "Uav, do you see me? establishing mutual attention between an uninstrumented human and an outdoor uav in flight," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015, pp. 3614–3620.

[74] M. Monajjemi, S. Mohaimenianpour, and R. Vaughan, "UAV, come to me: End-to-end, multi-scale situated HRI with an uninstrumented human and a distant UAV," in *Proceedings 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016, pp. 4410–4417.

[75] D. Murray and J. J. Little, "Using real-time stereo vision for mobile robot navigation," *Autonomous Robots*, vol. 8, pp. 161–171, 2000.

[76] T. Nikola, "Method of and apparatus for controlling mechanism of moving vessels or vehicles," US Patent 613 809, Nov. 8, 1898.

[77] W. S. Noble, "What is a support vector machine?" *Nature Biotechnology*, vol. 24, pp. 1565–1567, 2006.

[78] NVIDIA. "Jetson TK1 design diagram." Accessed 30-April-2017. (2015), [Online]. Available: `http://developer.download.nvidia.com/embedded/jetson/TK1/docs/242-7R375-1000-D00.Searchable.PDF.of.Assembly.Drawing.pdf`.

[79] NVIDIA. "Jetson TK1 embedded development kit." Accessed 30-April-2017. (2015), [Online]. Available: http://www.nvidia.ca/object/jetson-tk1-embedded-dev-kit.html.

[80] M. Olazaran, "A sociological study of the official history of the perceptrons controversy," *Social Studies of Science*, vol. 26, no. 3, pp. 611–659, 1996.

[81] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3400–3407.

[82] *Openrov*, [Accessed] August, 2019, May 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=OpenROV.

[83] *OptiTrack - Motion Capture System*, [Accessed] August, 2019, Nov. 2018. [Online]. Available: https://www.optitrack.com.

[84] Y. Pang, Y. Yuan, X. Li, and J. Pan, "Efficient HOG human detection," *Signal Processing*, vol. 91, pp. 773–781, 2011.

[85] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Association for Computational Linguistics*, Philadelphia, USA, 2002, pp. 311–318.

[86] M. Peikari, S. Salama, S. Nofech-Mozes, and A. L. Martel, "A cluster-then-label semi-supervised learning approach for pathology image classification," *Scientific Reports*, vol. 8, pp. 1–13, 2018.

[87] A. G. Perera, Y. Wei Law, and J. Chahl, "Uav-gesture: A dataset for uav control and gesture recognition," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Munich, Germany, 2018.

[88] K. Pfeil, S. L. Koh, and J. LaViola, "Exploring 3d gesture metaphors for interaction with unmanned aerial vehicles," in *International Conference on Intelligent User Interfaces*, Santa Monica, CA, USA: Association for Computing Machinery, 2013, pp. 257–266.

[89] PJRC. "Pjrc store." Accessed 30-April-2017. (2015), [Online]. Available: `https://www.pjrc.com/store/teensy32.html`.

[90] *Playstation Eye*, [Accessed] August, 2019, May 2019. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=PlayStation_Eye`.

[91] *Playstation Move*, [Accessed] August, 2019, Nov. 2018. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=PlayStation_Move`.

[92] E. Potokar, S. Ashford, M. Kaess, and J. G. Mangelson, "Holoocean: An underwater robotics simulator," in *International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, 2022, pp. 3040–3046.

[93] J. J. Prosser, H. V. Grey, and W. McKinnon, *Cave diving communications*. Cave Diving Section of the National Speleological Society, 1990. [Online]. Available: `http://www.swiss-cave-diving.ch/PDF-dateien/cavediving-signals.pdf`.

[94] L. Qu, S. Liu, X. Liu, M. Wang, and Z. Song, "Towards label-efficient automatic diagnosis and analysis: A comprehensive survey of advanced deep learning-based weakly-supervised, semi-supervised and self-supervised techniques in histopathological image analysis," *Physics in Medicine & Biology*, vol. 67, 20TR01, 2022.

[95] M. Quigley, K. Conley, B. Gerkey, *et al.*, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, Kobe, Japan, 2009.

[96] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, no. 2-3, pp. 709–730, 2020.

[97] A. V. Reddy, K. Shah, W. Paul, *et al.*, "Synthetic-to-real domain adaptation for action recognition: A dataset and baseline performances," in *IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023, pp. 11 374–11 381.

[98] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, pp. 779–788.

[99] L. Reiher, B. Lampe, and L. Eckstein, "A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–7.

[100] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," pp. 91–99, 2015. [Online]. Available: `http://arxiv.org/abs/1506.01497`.

[101] F. Rosenblatt, "The Perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, p. 386, 1958.

[102] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[103] N. S. Sanjay and A. Ahmadinia, "Mobilenet-tiny: A deep neural network-based real-time object detection for Rasberry Pi," in *IEEE International Conference On Machine Learning And Applications (ICMLA)*, Boca Raton, USA, 2019, pp. 647–652.

[104] G. Saporito, *What is a perceptron?* Sep. 2019. [Online]. Available: `https://towardsdatascience.com/what-is-a-perceptron-210a50190c3b`.

[105] M. W. Shelley, *Frankenstein; or, The Modern Prometheus*. Philadelphia, PA: Carey, Lea & Blanchard, 1833, www.gutenberg.org. Retrieved 24 Aug., 2023.

[106] P. Skalski, *How to use the segment anything model (sam)*, Jul. 2023. [Online]. Available: `https://blog.roboflow.com/how-to-use-segment-anything-model-sam/`.

[107] G. Stam and M. Ishino, Eds., *Integrating Gestures: The Interdisciplinary Nature of Gesture*. John Benjamins Publishing Company, 2011. [Online]. Available: `https://benjamins.com/catalog/gs.4`.

[108] B. Sundar and T. Bagyammal, "American sign language recognition for alphabets using mediapipe and LSTM," *Procedia Computer Science*, vol. 215, pp. 642–651, 2022.

[109] S. Suresh, M. H. TP, and M. Supriya, "Sign language recognition system using deep neural network," in *IEEE International Conference on Advanced Computing & Communication Systems (ICACCS)*, Coimbatore, India, 2019, pp. 614–618.

[110] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems (NEURIPS)*, vol. 27, 2014.

[111] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, 2017.

[112] *Tensorrt sdk | nvidia developer*, [Accessed] May, 2024, May 2024. [Online]. Available: https://developer.nvidia.com/tensorrt.

[113] M. Tomberlin, L. Tahai, and K. Pietroszek, "Gauntlet: Travel technique for immersive environments using non-dominant hand," in *IEEE Virtual Reality (VR)*, IEEE, Los Angeles, California, USA, 2017, pp. 299–300.

[114] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 4489–4497.

[115] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study," *Knowledge and Information systems*, vol. 42, pp. 245–284, 2015.

[116] A. M. Turk. "Amazon mechanical turk." Accessed 11-September-2018. (2018), [Online]. Available: https://www.mturk.com.

[117] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision (IJCV)*, vol. 104, pp. 154–171, 2013.

[118] J. Vertesi, *Seeing Like a Rover: How Robots, Teams, and Images Craft Knowledge of Mars*. The University of Chicago Press, 2015. [Online]. Available: https://www.press.uchicago.edu/ucp/books/book/chicago/S/bo18295743.html.

[119] B. Verzijlenberg and M. Jenkin, "Swimming with robots: Human robot communication at depth," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010, pp. 4023–4028.

[120] *Visual Signals*. US Army Publishing Directorate, Mar. 2017. [Online]. Available: https://arstechnica.com/wp-content/uploads/2017/03/visualsignalsmanual.pdf.

[121] J. Wang, S.-w. Luo, and X.-h. Zeng, "A random subspace method for co-training," in *IEEE International Joint Conference on Neural Networks*, Hong Kong, China, 2008, pp. 195–200.

[122] L. Wang, Y. Xiong, Z. Wang, *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *European Conference on Computer Vision (ECCV)*, Springer, Amsterdam, The Netherlands, 2016, pp. 20–36.

[123] *Wii*, [Accessed] August, 2019, Jul. 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wii.

[124] *Wii remote*, [Accessed] August, 2019, Nov. 2018. [Online]. Available: https://en.wikipedia.org/wiki/Wii_Remote.

[125] B. Wilcox and T. Nguyen, "Sojourner on mars and lessons learned for future planetary rovers," NASA's Jet Propulsion Laboratory, Tech. Rep., 1998.

[126] J. Xu, A. G. Schwing, and R. Urtasun, "Learning to segment under various forms of weak supervision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3781–3790.

[127] J. Yang, J. P. Wilson, and S. Gupta, "Diver gesture recognition using deep learning for underwater human-robot interaction," in *OCEANS*, Seattle, USA, 2019, pp. 1–5.

[128] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, "Image data augmentation for deep learning: A survey," *arXiv preprint arXiv:2204.08610*, 2022.

[129] L. Yao, A. Torabi, K. Cho, *et al.*, "Describing videos by exploiting temporal structure," in *IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 4507–4515.

[130] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, USA, 1995, pp. 189–196.

[131] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[132] F. Zhang, S. Chu, R. Pan, N. Ji, and L. Xi, "Double hand-gesture interaction for walk-through in VR environment," in *IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, IEEE, Seoul, South Korea, 2017, pp. 539–544.

[133] Y. Zhang, Y. Jiang, H. Qi, *et al.*, "An underwater human–robot interaction using a visual–textual model for autonomous underwater vehicles," *Sensors*, vol. 23, no. 1, p. 197, 2022.

[134] Z. Zhang, W. Mi, J. Du, *et al.*, "Design and implementation of a modular uuv simulation platform," *Sensors*, vol. 22, 2022.

[135] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, pp. 44–53, 2018.

[136] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[137] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[138] Y. Zhu, Z. Lan, S. Newsam, and A. Hauptmann, "Hidden two-stream convolutional networks for action recognition," in *Asian Conference on Computer Vision (ACCV)*, Springer, Perth, Australia, 2018, pp. 363–378.